

The copyright © of this thesis belongs to its rightful author and/or other copyright owner. Copies can be accessed and downloaded for non-commercial or learning purposes without any charge and permission. The thesis cannot be reproduced or quoted as a whole without the permission from its rightful owner. No alteration or changes in format is allowed without permission from its rightful owner.



**AN ADAPTIVE TRUST BASED SERVICE QUALITY
MONITORING MECHANISM FOR CLOUD COMPUTING**



MOHAMED FAZIL MOHAMED FIRDHOUS

**DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2016**

Permission to Use

In presenting this thesis in fulfillment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the University Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences

UUM College of Arts and Sciences

Universiti Utara Malaysia

06010 UUM Sintok

Abstrak

Pengkomputeran awan adalah paradigma terkini dalam pengkomputeran teragih yang menyediakan sumber pengkomputeran melalui Internet sebagai perkhidmatan. Oleh kerana daya tarikan pengkomputeran awan, pasaran kini dibanjiri oleh ramai pembekal perkhidmatan. Ini mewujudkan keperluan pelanggan untuk mengenal pasti pembekal perkhidmatan yang betul, yang akan memenuhi keperluan mereka dari segi kualiti perkhidmatan. Pemantauan kualiti perkhidmatan pengkomputeran awan sedia ada hanya terhad kepada pengukuran sahaja. Sebaliknya, peningkatan berterusan dan taburan skor kualiti perkhidmatan telah dilaksanakan dalam paradigma pengkomputeran teragih tetapi tidak khusus untuk pengkomputeran awan. Penyelidikan ini mengkaji kaedah-kaedah serta mencadangkan mekanisme untuk mengukur dan menentukan kedudukan kualiti perkhidmatan pembekal perkhidmatan. Penyelesaian yang dicadangkan dalam tesis ini terdiri daripada tiga mekanisme iaitu mekanisme perkhidmatan pemodelan kualiti, mekanisme pengkomputeran penyesuaian kepercayaan dan mekanisme pengedaran kepercayaan bagi pengkomputeran awan. Kaedah Penyelidikan Rekabentuk (KPR) telah diubah suai dengan menambah fasa, cara dan kaedah, dan hasil kemungkinan. KPR yang diubahsuai ini telah digunakan sepanjang kajian ini. Mekanisma ini telah dibangunkan dan diuji secara beransur-ansur sehingga mencapai hasil yang diharapkan. Satu set eksperimen yang menyeluruh telah dijalankan dalam persekitaran simulasi untuk mengesahkan keberkesanannya. Penilaian telah dijalankan dengan membandingkan prestasi mereka dengan gabungan model kepercayaan dan model kepercayaan QoS bersama-sama dengan mekanisme pengiraan kepercayaan berasaskan teori logik fuzi dan mekanisme pengagihan kepercayaan berasaskan konsep agen utama yang telah dibangunkan untuk sistem teragih lain. Keputusan menunjukkan mekanisme yang dicadangkan dalam tesis ini adalah lebih pantas dan stabil berbanding mekanisme sedia ada dalam mencapai skor kepercayaan akhir menggunakan kriteria yang diuji. Keputusan yang dibentangkan dalam tesis ini adalah penting dalam usaha untuk membolehkan pengguna mengesahkan prestasi pembekal perkhidmatan sebelum membuat pilihan yang tepat.

Kata kunci: Pengkomputeran awan, Pemantauan kualiti perkhidmatan, Pengkuantitian kualiti perkhidmatan, Pengkomputeran kepercayaan, Pengagihan kepercayaan

Abstract

Cloud computing is the newest paradigm in distributed computing that delivers computing resources over the Internet as services. Due to the attractiveness of cloud computing, the market is currently flooded with many service providers. This has necessitated the customers to identify the right one meeting their requirements in terms of service quality. The existing monitoring of service quality has been limited only to quantification in cloud computing. On the other hand, the continuous improvement and distribution of service quality scores have been implemented in other distributed computing paradigms but not specifically for cloud computing. This research investigates the methods and proposes mechanisms for quantifying and ranking the service quality of service providers. The solution proposed in this thesis consists of three mechanisms, namely service quality modeling mechanism, adaptive trust computing mechanism and trust distribution mechanism for cloud computing. The Design Research Methodology (DRM) has been modified by adding phases, means and methods, and probable outcomes. This modified DRM is used throughout this study. The mechanisms were developed and tested gradually until the expected outcome has been achieved. A comprehensive set of experiments were carried out in a simulated environment to validate their effectiveness. The evaluation has been carried out by comparing their performance against the combined trust model and QoS trust model for cloud computing along with the adapted fuzzy theory based trust computing mechanism and super-agent based trust distribution mechanism, which were developed for other distributed systems. The results show that the mechanisms are faster and more stable than the existing solutions in terms of reaching the final trust scores on all three parameters tested. The results presented in this thesis are significant in terms of making cloud computing acceptable to users in verifying the performance of the service providers before making the selection.

Keywords: Cloud computing, Service quality monitoring, Service quality quantification, Trust computing, Trust distribution

Declaration

Some of the works presented in this thesis have been published or submitted as listed below.

Book Chapters

[1] **Mohamed Firdhous**, Suhaidi Hassan, Osman Ghazali and Massudi Mahmuddin, "Evaluating Cloud System Providers: Models, Methods and Applications" in "Cloud Systems in Supply Chains" Chapter 7, Dr. Fawzy Soliman, Ed. (pp. 121-149). Basingstoke, Hampshire, UK: Palgrave Macmillan, 2014 (ISBN – 978-1-13-732425-2).

Journal Articles

[2] **Mohamed Firdhous**, Osman Ghazali, and Suhaidi Hassan, “A Mechanism for Distribution and Sharing of Trust Scores among Cooperating Cloud Computing Service Monitors”, Accepted for publication in Jurnal Teknologi (Sciences & Engineering) (Special Issue on Current and Emerging Trends in Technology, Science and Engineering), Scopus Indexed. (Won the best paper award at the Second AFAP Conference on Current and Emerging Trends in Science and Engineering, Surabaya, Indonesia 13/09/2014)

[3] **Mohamed Firdhous**, Osman Ghazali, and Suhaidi Hassan, “Modeling of Quality of Service based Trust for Cloud Computing”, Accepted for publication in Jurnal Teknologi (Sciences & Engineering) (Special Issue on Current and Emerging Trends in Technology, Science and Engineering), Scopus Indexed.

[4] **Mohamed Firdhous**, Osman Ghazali, and Suhaidi Hassan, “Robust Multi-Dimensional Trust Computing Mechanism for Cloud Computing”, Jurnal Teknologi (Sciences & Engineering) (Special Issue on Current and Emerging Trends in

Technology, Science and Engineering Vol. 2), vol. 69, no. 2, July 2014, pp. 1-6. Scopus Indexed.

[5] **Mohamed Firdhous**, Osman Ghazali, and Suhaidi Hassan, “Statistically Controlled Robust Trust Computing Mechanism for Cloud Computing”, Journal of Information and Communication Technology (JICT), vol. 13, 2014, pp. 23-39. Scopus Indexed.

[6] **Mohamed Firdhous**, Suhaidi Hassan and Osman Ghazali, “Monitoring, Tracking and Quantification of Quality of Service in Cloud Computing”, International Journal of Scientific & Engineering Research (IJSER), vol. 04, no. 05, May, 2013, pp. 112 – 117.

[7] **Mohamed Firdhous**, Suhaidi Hassan and Osman Ghazali, “A Comprehensive Survey on Quality of Service Implementations in Cloud Computing”, International Journal of Scientific & Engineering Research (IJSER), vol. 04, no. 05, May, 2013, pp. 118 – 123.

[8] **Mohamed Firdhous**, Suhaidi Hassan and Osman Ghazali, “Statistically Enhanced Multi-Dimensional Trust Computing Mechanism for Cloud Computing”, Journal of Mobile Computing and Multimedia Communications (IJMCMC), vol. 05, no. 02, April-June, 2013, pp. 1 – 17. Scopus Indexed.

[9] **Mohamed Firdhous**, Osman Ghazali, and Suhaidi Hassan, “Trust Management in Cloud Computing – A Critical Review”, International Journal on Advances in ICT for Emerging Regions (ICTer), vol. 04, no. 02, Sept. 2011, pp. 24 – 36.

Conference Papers

[10] **Mohamed Firdhous**, Suhaidi Hassan and Osman Ghazali, “Monitoring, Tracking and Quantification of Quality of Service in Cloud Computing” 3rd Global Conference for Academic Research on Scientific and Emerging Technologies (GCARSET), March 9–11, 2013, Kuala Lumpur, Malaysia, pp. 107-112.

[11] **Mohamed Firdhous**, Suhaidi Hassan and Osman Ghazali, “Quality of Service in Cloud Computing – A Critical Review” 3rd Global Conference for Academic Research on Scientific and Emerging Technologies (GCARSET), March 9–11, 2013, Kuala Lumpur, Malaysia, pp. 113-118.

[12] **Mohamed Firdhous**, Suhaidi Hassan and Osman Ghazali, “Hysteresis based Trust Computing Mechanism for Cloud Computing” 2012 IEEE Region 10 Conference (TENCON 2012), November 19–22, 2012, Cebu, Philippines, pp. 796-801.

[13] **Mohamed Firdhous**, Suhaidi Hassan, Osman Ghazali, and Massudi Mahmuddin, “Bio Inspired Trust Management in Distributed Systems - A Critical Review”, 2012 IEEE Conference on Open Systems (ICOS2012), October 21-24, 2012, Kuala Lumpur, Malaysia,

[14] **Mohamed Firdhous**, Suhaidi Hassan and Osman Ghazali, “Multi-Dimensional Trust Computing Mechanism for Cloud Computing” 3rd International Conference on Network Applications, protocols and services (NetApps 2012), September 19-20, 2012, Sintok, Kedah Darul Aman, Malaysia, pp. 7-12.

[15] **Mohamed Firdhous**, Osman Ghazali, and Suhaidi Hassan, “A Memoryless Trust Computing Mechanism for Cloud Computing”, The Fourth International Conference on Networked Digital Technologies (NDT'2012), April 24-26, 2012- Dubai, UAE, pp.174-185, published in Communications in Computer and Information Science, R.

Benlamri, Ed., Berlin Heidelberg, Germany: Springer-Verlag, 2012, Vol. 293, pp. 174–185. DOI: 10.1007/978-3-642-30507-8.

[16] **Mohamed Firdhous**, Osman Ghazali, and Suhaidi Hassan, "Applying Bees Algorithm for Trust Management in Cloud Computing", 6th International ICST Conference on Bio-Inspired Models of Network, Information, and Computing Systems (BIONETICS 2011), December 5th - 7th, 2011 - York, England, published in Lecture Notes on ICST (LNICST) E. Hart et al. Eds., Berlin Heidelberg, Germany: Springer-Verlag, 2012, Vol. 103, pp. 224–229.

[17] **Mohamed Firdhous**, Osman Ghazali, and Suhaidi Hassan, "A trust computing mechanism for cloud computing with multilevel thresholding", Sixth IEEE International Conference on Industrial & Information Systems (ICIIS2011), August 16-19, 2011, Kandy, Sri Lanka, pp. 457-461.

[18] **Mohamed Firdhous**, Osman Ghazali, and Suhaidi Hassan, "A trust computing mechanism for cloud computing", 4th ITU Kaleidoscope Academic Conference, December 12–14, 2011, Cape Town, South Africa, pp. 199-205.

[19] **Mohamed Firdhous**, Osman Ghazali, Suhaidi Hassan, "Modeling of cloud system using Erlang formulas", 17th Asia-Pacific Conference on Communications, October 3-5, 2011, Kota Kinabalu, Sabah, Malaysia, pp. 411-416.

[20] **Mohamed Firdhous**, Osman Ghazali, Suhaidi Hassan, Nur Ziadah Harun and Azizi Abas, "Honey Bee Based Trust Management System for Cloud Computing", 3rd International Conference on Computing and Informatics (ICOCI2011), June 8-9, 2011 Bandung, Indonesia, pp. 327-332.

Acknowledgments

In the name of Allah the Most Beneficent, the Most Merciful.

Studying towards a PhD, though it is very rewarding at the end, it is a very tedious and challenging journey. I have almost reached the end of it with a lot of effort, blood, sweat and tears. There are several people, who helped to reach this stage successfully. I will fail in my duty, if I do not give them the credit that is rightfully due to them.

First and foremost, I must thank my supervisors Prof. Madya Dr. Osman Ghazali and Prof. Dr. Suhaidi Hassan. It has been an honor to be a PhD student under them. I really appreciate all their contributions in terms of advice, support, time and ideas that made my PhD experience productive and stimulating. A special thanks for encouraging me to write research articles and proceeding papers that helped me a lot in improving my language skills and would go a long way in my career development as an academic.

I would like to extend my sincere gratitude to my former vice chancellor Prof. Malik Ranasnghe at the University of Moratuwa, Sri Lanka. He not only encouraged me to start my PhD, but also give the list of people who I should approach along with a strong recommendation letter. He also approved my study leave at the shortest possible time.

Then I would like to thank my InterNetWoks research laboratory members both the academic staff and students. They kept the environment very nice and stimulating throughout my stay there.

Finally I would extend my gratitude to my family especially my wife Shamila. I really appreciate her understanding and support. I really have to mention my daughter Fathima Sameeha and son Mohamed Shabaz, though they missed me a lot during this period, they gave me the reason and courage that I should finish this journey successfully.

Table of Contents

Permission to Use	ii
Abstrak	iii
Abstract	iv
Declaration	v
Acknowledgments	ix
Table of Contents	x
List of Tables	xiv
List of Figures	xv
List of Abbreviations	xviii
 CHAPTER ONE OVERVIEW	 1
1.1 Introduction	1
1.2 Background	1
1.2.1 Cloud Computing	2
1.3 Problems and Issues Pertaining to Cloud Computing	5
1.4 Research Motivation	7
1.5 Problem Statement	9
1.6 Research Questions	11
1.7 Research Objectives	11
1.8 Research Scope	12
1.9 Significance of the Research and Expected Contributions	13
1.10 Organization of the Thesis	14
 CHAPTER TWO LITERATURE REVIEW	 16
2.1 Introduction	16
2.2 Cloud Computing	16
2.2.1 Cloud Computing Service Offerings	18
2.2.2 Cloud Computing Deployment Models	21
2.3 Quality of Service in Cloud Computing	27
2.3.1 Service Quality Monitoring in Cloud Computing	34

2.3.2	Service Quality Parameters in Cloud Computing	39
2.3.3	Definition of Performance Metrics	42
2.4	Trust Computing and Management	44
2.4.1	Trust Management in Cloud Computing	48
2.5	Summary	61
CHAPTER THREE RESEARCH METHODOLOGY		62
3.1	Introduction	62
3.2	Research Approach	64
3.2.1	Analysis	66
3.3	Design	70
3.3.1	Model Development	72
3.3.2	Model Implementation	73
3.3.3	Model Validation	74
3.4	Testing	76
3.5	Evaluation	79
3.5.1	Selecting the Evaluation Approach	79
3.5.2	CloudSim Simulation Suite	86
3.5.3	Experiment Environment	90
3.6	Summary	97
CHAPTER FOUR SERVICE QUALITY MODELING MECHANISM FOR CLOUD COMPUTING		99
4.1	Introduction	99
4.2	Normalizing of Performance Metrics	100
4.3	Modeling of Service Quality of Cloud Providers	103
4.3.1	Single Parameter Service Quality Quantification Mechanism (SP-SQQM)	104
4.4	Multi-Parameter Service Quality Quantification Mechanism (MP-SQQM)	107
4.4.1	Computing Trust Score with Different Priorities	111
4.5	Functional Verification of MP-SQQM	112
4.6	Summary	115

CHAPTER FIVE	ADAPTIVE TRUST COMPUTING MECHANISM	
	FOR CLOUD COMPUTING	116
5.1	Introduction	116
5.2	Trust Formation and Evolution	117
5.3	Adaptive Continuous Trust Evolution Mechanism (ACTEM)	118
5.3.1	Functional Verification of ACTEM	120
5.4	Memoryless Trust Computing Mechanism (MemTrust)	122
5.4.1	Functional Verification of MemTrust	125
5.5	Hysteresis-based Trust Evolution Mechanism (HystTrust)	127
5.5.1	Hysteresis Function	129
5.5.2	Pseudo Code of the Proposed Algorithm	130
5.5.3	Functional Verification of HystTrust	130
5.6	Robust Adaptive Trust Computing Mechanism (RATComM)	132
5.6.1	Functional Evaluation of RATComM	133
5.7	Multi-Dimensional Trust Computing Mechanism (MuDTComM)	136
5.7.1	Functional Evaluation of MuDTComM	139
5.8	Summary	140
CHAPTER SIX	PROBABILITY-BASED TRUST DISTRIBUTION	
	MECHANISM FOR CLOUD COMPUTING	141
6.1	Introduction	141
6.2	Distribution of Trust Scores	142
6.2.1	Trust Table Updating Process	144
6.3	Probability-based Trust Distribution Mechanism (PTDiMech)	146
6.3.1	Functional Evaluation of PTDiMech	150
6.4	Summary	151
CHAPTER SEVEN	PERFORMANCE ANALYSIS OF TRUST	
	COMPUTING AND DISTRIBUTION MECHANISMS	153
7.1	Introduction	153
7.2	Simulation Environment	153
7.3	Performance Analysis of Service Quality Quantification Mechanisms . . .	156
7.3.1	Performance Analysis of SP-SQQM	157

7.3.2	Performance Analysis of MP-SQQM	160
7.4	Performance Analysis of Trust Computing Mechanisms	162
7.4.1	Performance Analysis of Adaptive Continuous Trust Evolution Mechanism	164
7.4.2	Performance Analysis of MemTrust	167
7.4.3	Performance Analysis of HystTrust	168
7.4.4	Performance Analysis of RATComM	170
7.4.5	Performance Analysis of MuDTComM	171
7.5	Performance Analysis of Trust Distribution Mechanism	173
7.6	Summary	179
 CHAPTER EIGHT CONCLUSIONS AND FUTURE WORK		182
8.1	Introduction	182
8.2	Summary of Research	182
8.3	Research Contributions	185
8.4	Research Limitations	186
8.5	Recommendations for Future Work	187
 REFERENCES		189

List of Tables

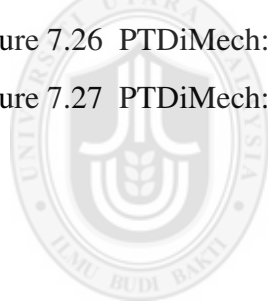
Table 2.1	Summary of Features of the Service Quality Monitoring Mechanisms	37
Table 2.2	Summary of Trust Computing Mechanisms for Cloud Computing	58
Table 3.1	Comparison of Different Evaluation Approaches	80
Table 3.2	Comparison of Different Cloud Simulators	86
Table 3.3	Virtual Machine Mapping	93
Table 3.4	Experiment Setup Attributes and Values	94
Table 4.1	Performance Table	110
Table 4.2	Modified Performance Table	112
Table 6.1	Sample Trust Table	146
Table 6.2	Conditional Probability Table at N_S for N_D	149
Table 7.1	Specification of the Host Computer	154
Table 7.2	Experiment Setup Attributes for the Evaluation of SQQMs	156
Table 7.3	Service Quality Requirements for Service Quality Quantification Mechanisms	156
Table 7.4	Experiment Setup Attributes for the Evaluation of Trust Computing Mechanisms	163
Table 7.5	Service Quality Requirements for Trust Computing Mechanisms	163
Table 7.6	Experiment Setup Attributes for the Evaluation of Trust Distribution Mechanism	173
Table 7.7	Service Quality Requirements for Trust Distribution Mechanism	174

List of Figures

Figure 1.1	Capacity Utilization Curve	3
Figure 2.1	Cloud Computing Service Offerings	19
Figure 2.2	Cloud Computing Deployment Models	22
Figure 3.1	Research Methodology	63
Figure 3.2	Stages of Design Research Methodology	64
Figure 3.3	Research Approach	66
Figure 3.4	Main Steps Involved in Research Clarification Stage	67
Figure 3.5	Main Steps in Descriptive Study - I	69
Figure 3.6	Conceptual Model	70
Figure 3.7	Mechanism Development Process	71
Figure 3.8	Eclipse Integrated Development Environment for Java	77
Figure 3.9	QJ-Pro Code Analysis Window	78
Figure 3.10	CloudSim Layered Architecture	88
Figure 3.11	CloudSim Class Diagram	89
Figure 3.12	Simulation Steps	91
Figure 4.1	Change in Trust Scores	105
Figure 4.2	Comparative Change in Trust Scores	106
Figure 4.3	Naive Bayesian Network	109
Figure 4.4	Naive Bayesian Network for a Cloud Computing System	109
Figure 4.5	Trust Score Computed Using Two Input Parameters	113
Figure 4.6	Effect of Weights on Trust Scores Computed	114
Figure 4.7	Effect of Weights on Final Trust Score	114
Figure 5.1	Trust Management System	119
Figure 5.2	Trust Scores due to Continuous Positive or Negative Feedbacks	121
Figure 5.3	Effect of Confidence Level on Trust Scores Computed	121
Figure 5.4	MemTrust Trust Evolution Unit	122
Figure 5.5	Sigmoid Function	124

Figure 5.6	Modified Sigmoid Function	125
Figure 5.7	Trust Scores Computed for Constant Positive Responses	126
Figure 5.8	Trust Scores Computed for Constant Negative Responses	126
Figure 5.9	Trust Scores Computed for Random Response Time Requirement	127
Figure 5.10	MemTrust Trust Evolution Unit	128
Figure 5.11	Sample Hysteresis Curve	129
Figure 5.12	Comparison of Trust for Random Response Times	132
Figure 5.13	RATComM Trust Evolution Unit	133
Figure 5.14	Trust Scores with 90% Validated Inputs Vs. Non Validated Inputs	135
Figure 5.15	Trust Scores with 95% Validated Inputs Vs. Non Validated Inputs	135
Figure 5.16	Effect of Confidence Level on Trust Scores	136
Figure 5.17	MuDTComM Trust Evolution Unit	137
Figure 5.18	MuDTComM Trust Evolution Unit in Detail	137
Figure 5.19	Comparison of RATComM and MuDTComM Trust Evolution Units	139
Figure 5.20	The Effect of Weights and Confidence Level on Trust Scores	140
Figure 6.1	High-Level Architecture of Trust Distribution System	143
Figure 6.2	Trust Administration Unit	144
Figure 6.3	Trust Updating Process	145
Figure 6.4	Bayesian Network for Node NS	148
Figure 6.5	Change of Trust Scores for CSP_1 over a Period of Time	151
Figure 7.1	Eclipse IDE Loaded with CloudSim	154
Figure 7.2	A Typical CloudSim Life Cycle	155
Figure 7.3	SP-SQQM: Trust Scores Computed Using Response Time	158
Figure 7.4	SP-SQQM: Trust Scores Computed Using Service Time	159
Figure 7.5	SP-SQQM: Trust Scores Computed Using Availability	159
Figure 7.6	MP-SQQM: Trust Scores with Equal Weights	160
Figure 7.7	MP-SQQM: Trust Scores with Unequal Weights (Case I)	161
Figure 7.8	MP-SQQM: Trust Scores with Unequal Weights (Case II)	161
Figure 7.9	ACTEM: Trust Scores Computed based on Response Time	165
Figure 7.10	ACTEM: Trust Scores Computed based on Service Time	166
Figure 7.11	ACTEM: Trust Scores Computed based on Availability	166

Figure 7.12	MemTrust: Trust Scores Computed based on Response Time	167
Figure 7.13	MemTrust: Trust Scores Computed based on Service Time	167
Figure 7.14	MemTrust: Trust Scores Computed based on Availability	168
Figure 7.15	HystTrust: Trust Scores Computed based on Response Time	169
Figure 7.16	HystTrust: Trust Scores Computed based on Service Time	169
Figure 7.17	HystTrust: Trust Scores Computed based on Availability	169
Figure 7.18	RATComM: Trust Scores Computed based on Response Time	170
Figure 7.19	RATComM: Trust Scores Computed based on Service Time	170
Figure 7.20	RATComM: Trust Scores Computed based on Availability	171
Figure 7.21	Trust Scores Computed by MuDTComM and Fuzzy Mechanisms .	172
Figure 7.22	Effect of Weights on Trust Scores - MuDTComM vs Fuzzy	172
Figure 7.23	PTDiMech: Trust Scores Computed based on Response Time	176
Figure 7.24	PTDiMech: Trust Scores Computed based on Service Time	177
Figure 7.25	PTDiMech: Trust Scores Computed based on Availability	178
Figure 7.26	PTDiMech: Trust Scores Computed with Equal Weights	178
Figure 7.27	PTDiMech: Trust Scores Computed with Different Weights	179

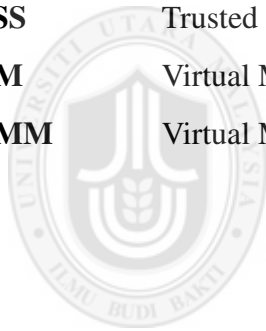


UUM
Universiti Utara Malaysia

List of Abbreviations

ACTEM	Adaptive Continuous Trust Evolution Mechanism
API	Application Programming Interface
AWS	Amazon Web Services
CDO	Cloud Deployment Options
CSP	Cloud Service Provider
DRM	Design Research Methodology
DS-I	Descriptive Study I
DS-II	Descriptive Study II
FBCT	Family-gene Based model for Cloud Trust
FIFO	First In First Out
GUT	Graphical User interface
HystTrust	Hysteresis-based Trust Evolution Mechanism
IaaS	Infrastructure as a Service
IdP	Identity Policy
IdPS	Identity Practice Statement
IDE	Integrated Development Environment
ISO	International Standards Organization
JVM	Java Virtual Machine
KPI	Key Performance Indicators
MemTrust	Memoryless Trust Computing Mechanism
MP-SQQM	Multi-Parameter Service Quality Quantification Mechanism
MTCEM	Multi-tenancy Trusted Computing Environment Model
MuDTComM	Multi-Dimensional Trust Computing Mechanism
PaaS	Platform as a Service
PERMIS	PrivilEge and Role Management Infrastructure Standard
PS	Prescriptive Study
PSO	Particle Swarm Optimization
PTDiMech	Probability-based Trust Distribution Mechanism

QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RATComM	Robust Adaptive Trust Computing Mechanism
RC	Research Clarification
S3	Simple Storage Service
SaaS	Software as a Service
SLA	Service Level Agreement
SMI	Service Measurement Index
SP	Service Policy
SPS	Service Practice Statement
SP-SQQA	Single Parameter Service Quality Quantification Algorithm
SP-SQQM	Single Parameter Service Quality Quantification Mechanism
TSS	Trusted Platform Software Stack
VM	Virtual Machine
VMM	Virtual Machine Manager



UUM
Universiti Utara Malaysia

CHAPTER ONE

OVERVIEW

1.1 Introduction

This chapter presents a brief introduction to the proposed research along with the general background information on cloud computing in brief including its features, advantages, disadvantages and issues. The chapter also outlines the problem statement and research questions, research motivation, research objectives, research scope and the significance of the research along with the contributions. Finally the outline of the proposal is presented at the end.

1.2 Background

Cloud computing has become very popular among the computing community in the recent years. It has already earned the nickname the *5th utility* due to its versatile and economic way of making resources available over the Internet [1]. Utilities make the resources available to a wider clientele and charge them only for the usage. Electricity, water, gas and telephony are the four major utilities that have been commonly used in this manner before the arrival of cloud computing. Prior to the emergence of cloud computing in the latter part of the 1st decade of 2000s, computing resources such as hardware including processor power, storage, networks bandwidth were either purchased outright and installed in the data centers owned and operated by end users themselves or leased from public data centers on fixed monthly or annual charges [2]. The clients installed the operating systems, tools and applications of their choice on these hardware dedicated only for their use. Once the hardware has been purchased or leased in this manner, the capacity of these systems were fixed irrespective of usage. The computing resources thus installed in clients' data centers are generally underutilized. Recent surveys have found that in many data centers the

server utilization has been between 10% and 30% of their available computing power and less than 5% in desktop computers [3]. On the other hand, underspending on computing resources would affect business performance especially during peak times due to lack of resources [4]. Both under utilization as well as over exploitation result in losses due to wastage and loss of business respectively. Hence an ideal solution would be the right sizing of the required resources with the capability to expand and shrink to suit the demand patterns. Cloud computing systems can closely follow the demand patterns during both peak and off-peak hours [5]. Hence cloud computing offers the right platform for clients to host their services as it provides them with the right computing solution that responds to their performance requirements at the lowest possible cost.

1.2.1 Cloud Computing

Cloud computing has transformed the computing market from a product based one to a service oriented one. Under cloud computing, both hardware and software resources including computing power, operating systems and development platforms and user applications have been made available for customers over the Internet to access and pay for only the services received [1]. Prior to the appearance of cloud computing, customers either purchased the hardware and installed them in house or leased from public data center operators at fixed charges. Thus the investment on computing resources was considered capital expenditure that did not follow the real usage patterns. Figure 1.1 shows the capacity vs utilization curve developed by Amazon Web Services (AWS) for storage requirements under the traditional purchase/own or lease and cloud computing models [6]. From Figure 1.1, it can be seen that the actual utilization of resources undergo heavy fluctuations in short as well as in the longer runs. The short term fluctuations represent the changes in demand due to time of day variations, where the demand peaks during working hours, subsiding towards

evening and in the total idle state in the late hours such as nights. The long term changes include the seasonal patterns and then the permanent increase and decrease in demands. Traditional computing model are unable to follow these changing demand patterns resulting in the investment on computing resources sub-optimal. On the other hand, cloud hosted systems can follow the demand patterns very closely even responding positively to the short or long term fluctuations.

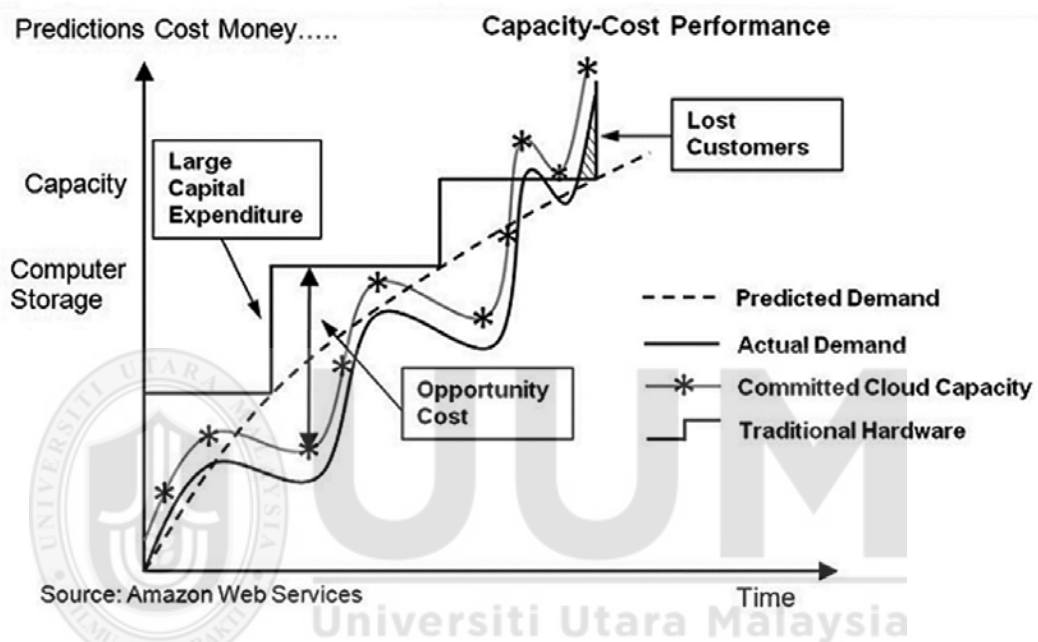


Figure 1.1. Capacity Utilization Curve [6]

For both start up as well as established organizations, the investment in computing resources up-front creates a burden by reducing the available financial capital that can be invested in their core business activities. Due to this limitation they might either reduce the size of their operations or the capacity of computing systems. Doing any of the above would affect the performance of the business due to lack of resources. Generally, businesses commencing afresh would start small and grow larger with time [7]. The growth patterns of these businesses at the beginning are unpredictable due to many reasons including the customer acceptance, competition from other businesses, general economic conditions of the country etc. If all the computing requirements for the business are purchased and hosted in public cloud services, the usage patterns as

well as the investment on them would be totally synchronized with the real resource demands. Also, the capital expenditure on computing resources would be minimal as the clients are required to pay for the usage only like in telecommunications, electricity, gas and water. In short, through cloud computing, it is possible to achieve a complete transformation the way computing resources are accessed and paid for. Since capital expenditure on the computing resources have been eliminated, it is possible to invest that money on other core business activities [8].

The underpinning technology that makes cloud computing possible is hardware virtualization [9]. Cloud computing resources especially the hardware are hosted on virtualized systems [10]. Hardware virtualization partitions the hardware resources logically into several time-shared or space-multiplexed units [11, 12]. These logical units are then presented to the customers as fully fledged systems that can mimic the entire physical system providing controlled access to the physical and logical resources.

Cloud service offerings are currently divided into three main groups, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [13]. The cloud service offerings are explained in detail in Chapter Two. In order to make the different service offerings possible, there requires to certain underlying physical infrastructure. This physical infrastructure is made up of the actual physical computing hardware along with a virtualization software installed on it. The physical hardware is the real workhorse that carries out the processing and other work such as temporary and permanent storage of data and application code, communication with other systems etc. The physical hardware is generally provided in the form of computing clusters, grids or individual servers [14]. The virtualization software commonly known as the Virtual Machine Manager (VMM) creates multiple on the fly

virtual computers on top of the same hardware [15]. The VMM provides the necessary isolation and security between the multiple virtual machines running in parallel on a single physical computer.

Thus cloud computing provides many advantages to both service providers and customers [16]. From the service providers perspective, they can improve the utility and productivity of their hardware resources. Improving the utility of the systems results in increased profitability per system. On the customers side, they can drastically reduce the cost of computing services. The customers are no longer required to invest on the computing resources upfront. This makes the computing resources as operational expenditure rather than a capital expenditure [17]. Also they are required to pay only for the usage, this makes the investment on computing optimal as there is no waste of resources. Since the entire processing hardware and software are installed and maintained at the service providers site, the cost of human resources required for the management of these resources is also reduced to the minimum. Presently the requirement of Computing professionals is limited to desktop support for helping the users with simple day-to-day problems. High end professional services such as installing and managing server level services will be taken care of by the cloud service provider.

1.3 Problems and Issues Pertaining to Cloud Computing

Though cloud computing provides many advantages to both customers and service providers in terms of cost savings and utilizations, it still needs to earn the confidence of the customers in certain other aspects for it to become commonly deployed and successful technology [18]. The dynamic nature of cloud computing as a result of the creation and hosting of virtual systems on the fly makes the performance of cloud systems unpredictable [19]. Many of the commercial applications including multi-

tiered business applications, scientific data processing, multimedia applications that can benefit from cloud computing are highly sensitive to quality variations [20]. The service quality requirements to be met by service providers along with the penalties to be imposed, in case of violation are specified in Service Level Agreements (SLA) signed by the parties [21]. An example SLA to be signed between Amazon Web Services (AWS) a leading cloud provider and its customers who wish to use its Amazon Simple Storage Service (S3) can be found in [22]. As per the SLA, the AWS commits to take commercially reasonable efforts to maintain the availability of Amazon S3 at least at 99.9 percent during any monthly billing cycle. The compensation for failing to meet the above commitment is service credit, which is also described in the SLA. There is no further commitment made on any other service quality expectations.

From the above discussion, it can be seen that the commitments made by leading cloud service providers at present are too simple and does not consider the complex application specific requirements. This kind of SLAs may not be strong enough to attract business customers whose applications are more sensitive to fluctuations in service quality in more than one dimensions (parameters) [23]. The situation has been more aggravated by the news item reported in the media about the high-profile crash of Amazon EC2 cloud services in 2011 [24]. This service outage affected many high profile businesses, who had hosted their services at AWS. Not only the site was down for many days and but also some organizations lost their data permanently.

Hence the cloud service providers need to come up with innovative methods to provide the service quality demanded by different types of applications and also to assure them that these commitments will be maintained. Also there should be independent monitoring and verification of the claims by service providers that they have met

commitments made to the customers satisfactorily. Only if the above can be provided, customers will have confidence on the service providers and would readily move their applications to cloud systems in order to reap the benefits of cloud computing. If an independent mechanism has been developed and made available for customers to check and verify the service quality of different cloud service providers based on customers' requirements and providers' prior performance before signing the SLA, the violations of the commitments by the service providers can be minimized.

1.4 Research Motivation

The cloud computing provides very attractive proposition for both service providers and customers alike at the same time. But people will take sometimes before they accept cloud computing as their primary choice for handling all their computing needs [3]. Especially the business community needs to be sure of many things including security, performance, reliability, availability etc., before outsourcing all their computing needs to a public cloud service provider [25, 26]. The performance of computers is a vital factor for operational sustainability as businesses become more and more dependent on computers for reaching a larger customer base [27, 28]. This is mainly due to the reason that the service quality parameter values are stochastic and vary significantly due unpredictable workloads, hardware and software malfunction etc. This necessitates the need for knowing the current status of the system in terms of its capability to meet the required service quality targets [29].

Hence the businesses need to continuously monitor the performance of their computing resources [30, 31]. When business applications are moved to the cloud, the need for monitoring them becomes more critical than before as the systems are hosted remotely in a data center owned and operated by a third party service provider. Many leading researchers in this field have also expressed the same sentiments on this

issue. For example, Garg et al. in [32] state that an effective service quality monitoring mechanism is essential for making cloud computing acceptable to wider spectrum of users. Ward and Baker in [33] have observed that monitoring is an important aspect of large scale distributed systems including cloud computing. Mohamaddiah et al. have carried out a survey of resource allocation and monitoring techniques deployed in cloud computing systems [34]. But the monitoring discussed in this article is limited to that of the service providers monitoring themselves for efficient allocation of resources to increase the productivity of their systems. The monitoring system proposed by Brinkmann et al. concentrates on monitoring the clouds for the purpose of billing [35]. They mainly concentrate on monitoring the system logs for obtaining data for this purpose. Suakanto et al. have carried out an experiment to measure service quality of cloud hosted web applications [36]. The experiments have used two parameters namely, response time and the number requests timed-out (availability) to understand the performance of cloud systems under increasing number of users. They finally conclude that as the number users (requests) increase the service quality decreases in terms of response time and availability. Despite the widespread popularity enjoyed by cloud computing, cloud monitoring has not been received much attention from the research community [37, 33]. This leaves a big gap that needs the attention of the research community urgently.

Many researchers have approached the service quality issue from the service providers perspective. The proposed mechanisms mainly concentrate on optimum resource allocation while meeting the SLA signed with the customer with the objective of revenue maximization [21, 38, 39, 40, 41, 42]. On the other hand, monitoring cloud services from customers' perspectives is also important. The monitoring mechanisms that have been currently used in the cloud computing arena have been an ad hoc collection of data collection, analysis, reporting, automation and decision

making software developed for other distributed computing paradigms such as high performance computing, grid computing and cluster computing [33]. Though there are similarities between some of the previous distributed system paradigms and cloud computing in terms of size and scale of operations and scalability, there are certain features such as virtualization and elasticity are unique to cloud computing only [43, 44].

The above mentioned reasons played the role of catalyst in motivating to start a research for developing mechanisms that can be used for monitoring and ranking of cloud service providers based on their service quality.

1.5 Problem Statement

Before cloud computing becomes the primary choice of users for all their computing needs many issues need to be addressed and solved [45]. Some of these issues that need the immediate attention of the researchers and developers include the assurance of service quality, energy efficiency, security and trust. Computers have become a strategic tool for many businesses [46]. Computers now control the core activities of many businesses. Hence prior to these businesses outsource all their computing requirements to public cloud services, they need to be satisfied by the service quality they receive as it will decide the performance of the core business operations. Service quality of a given cloud system is not static as it will change over time due to many factors such as the installed capacity, number of concurrent users and the demand placed by each user application on the system [38]. Hence, prior to entering into an agreement with service providers, customers need to be satisfied with the ability of the provider's ability to meet their demands [47]. Thus monitoring of cloud systems for the service quality from the customers' perspectives become imperative. Users not only need the ability to identify the service quality of service providers but also rank

them accordingly based on it [32]. Though the need for the independent monitoring of cloud services has already been identified, it is still an under researched area [33].

The QoS trust model, combined trust model and FIFO trust model have recently been proposed for monitoring and quantification of cloud service quality using trust as the parameter [48, 49]. However, these mechanisms do not address the complete picture of monitoring and quantifying large cloud services spread over a large geographical area from an independent monitor's perspective. Moreover, these mechanisms have not taken the dynamic nature of cloud system performance into account when arriving at the final trust scores computed using the service quality parameters. The current trust computing mechanisms deployed in the distributed computing arena also suffer from certain shortcomings. The trust computing mechanism proposed by Gu et al. [50] for cloud computing does not take the dynamic nature of the cloud systems into account. Due to the customers arriving and leaving dynamically at a cloud system, the performance of the system may exhibit momentary fluctuations. If the fluctuations are contained within predetermined limits, the performance may still be considered acceptable. This criterion must be included when the trust of a dynamic cloud system is to be computed. The monitoring and trust computing mechanisms proposed by Manuel [49] and Gu et al. [50] have a very limited reach geographically as they are not distributed in nature. The super agent based reputation management system for decentralized systems proposed by Wang et al. in [51] does not include a mechanism for identifying the performance differences experienced by customers located in different geographical regions. This is an important factor that must be taken into account in computing trust of systems that serves a large geographically distributed set of clients.

Therefore developing a complete monitoring mechanism capable of monitoring,

tracking, quantifying and distributing the dynamic service quality of cloud providers is of essence. Thence the overall aim of this research is to propose mechanisms that can be used to monitor, track, quantify and distribute the dynamic service quality of cloud providers from the customers' perspective.

1.6 Research Questions

In order to achieve the said purpose, the work has been organized in such a manner to find answers to the following research questions.

- Q1:** How the service qualities of different cloud service providers can be monitored, quantified and ranked?
- Q2:** What are the methods that can be used for adjusting the service quality scores dynamically to reflect the current performance of the service providers?
- Q3:** How can the monitoring mechanisms proposed in answering Q1 and Q2 be made to cover a larger geographical area?

1.7 Research Objectives

The aim of this research is to come up with mechanisms to monitor, track, quantify and distribute the dynamically changing the service quality of cloud service providers. The proposed mechanisms must be able to track the performance in many dimensions using multiple service quality parameters and quantify them in an easily understandable form as a single score. In order to cover a large geographical area, the proposed mechanisms must be capable of using multiple monitors and sharing the scores between them. This aim could be further explained with the aid of the following specific research objectives.

- O1:** To develop mechanisms for monitoring and quantifying the service quality of cloud service providers based on either a single parameter or multiple parameters.
- O2:** To propose mechanisms for adjusting the service quality scores known as the trust scores in response to the continued changes in performance of the service providers.
- O3:** To come up with a mechanism for distributing and sharing the trust scores between cooperating monitors with the aim of covering a larger geographical service area for including many service providers.

1.8 Research Scope

The overall goal of this research is to develop mechanisms for monitoring and quantifying the service quality of cloud service providers. The proposed mechanism must support any customer to identify the right service provider, who could meet their service requirements and rank them based on their performance. The customers must be able to provide the monitor with a set of service quality parameters and the expected values for them. Then by using the mechanisms proposed in this research, the monitor must be able to quantify the service quality of the service providers in the region and rank them accordingly for the purpose of identifying the most suitable service provider for the customer.

Further, when quantifying the service quality in this research, only the service quality of the infrastructure providers has been considered. Thus, the mechanisms proposed in this research are more suitable for monitoring infrastructure services. The monitoring and quantification of other services such as PaaS and SaaS are left for future works.

It must also be mentioned that the scope of this work is limited to monitoring only. No

attempt has been made to improve the service quality of providers or come up with mechanisms or algorithms that would help service providers to improve their services nor meet the customers' requirements.

1.9 Significance of the Research and Expected Contributions

This research presented several mechanisms that can be used to monitor, quantify, adjust and distribute service quality score known as the trust score of cloud service providers. The proposed mechanisms are capable of monitoring the performance of cloud systems continuously and adjusting the service quality (trust) scores proactively based on their most recent performance. The proposed mechanisms, if implemented by independent cloud service intermediaries such as cloud service brokers can help the prospective customers to identify right service providers based on their service quality requirements. Cloud service providers can also benefit from these mechanisms. if implemented internally by detecting the service quality degradations and take the remedial actions long before they become serious problems. With the aid of the probability-based trust distribution mechanisms, customers can identify the right service providers operating in a different geographical area than that of the customers. Thus, the mechanisms proposed in this research can help multiple stake holders including customers, service providers and cloud intermediaries such as brokers, exchange owners and clearing houses. The contributions of this research are summarized as follows:

- C1:** Mechanisms for service quality quantification based on either a single parameter or multiple parameters.
- C2:** Mechanisms for modifying the service quality scores known as the trust scores based on the continued performance of the service providers.
- C3:** A mechanism for distributing and sharing the trust scores between cooperating

monitors to cover the wider service area.

1.10 Organization of the Thesis

This thesis has been organized into eight chapters. Chapter One provides an overview of the overall work including the outlines of the research motivation, problem statement and research questions, research objectives, scope of the research and the significance of the research along with the contributions.

Chapter Two critically evaluates and summarizes the literature relevant to the topic of the study. The chapter provides the background information on cloud computing along with discussion on the recent published work. The chapter also includes an in depth discussion on the service quality monitoring and implementations in cloud computing. Finally an in depth analysis is presented on the trust computing mechanisms proposed for distributed systems.

Chapter Three establishes the research methodology adopted in this research work. The design research methodology has been adapted to suit the requirements of this work. The details of every step along with different approaches used within those steps have been explained in detail.

Chapter Four presents the service quality quantification mechanisms proposed in this research. The chapter service quality of cloud computing in detail along with mathematical equations for a selected set of parameters that are commonly used for measuring service quality in real time systems. Then the chapter discusses how service quality of cloud computing can be modeled from customers' perspectives. Finally the chapter describes two service quality quantifications named SP-SQQM and MP-SQQM, which quantify service quality based on a single parameter and multiple

parameters respectively.

Chapter Five discusses the trust computing mechanisms proposed in this research in detail. The chapter presents how the two main trust computing mechanisms Robust Adaptive Trust Computing Mechanism (RATComM) and Multi-Dimensional Trust Computing Mechanism (MuDTComM) have been designed in a step by step fashion. All the intermediate mechanisms that contributed to the final contributions were explain in detail in this chapter.

Chapter Six explains the probability-based Trust Distribution Mechanism that help share trust scores between cooperating monitoring and quantification nodes. The mechanism has been developed from basic principles of conditional probability and based on Bayes' Theorem as the basis for accounting for the differences between the performance values received and the perceived quality received by users.

Chapter Seven is concerned about the performance evaluation of all the mechanisms proposed in this research. The chapter discusses the simulation environment used for the evaluation of the mechanisms along with the benchmarking mechanisms that have been used for validating the proposed work.

Chapter Eight concludes this thesis by summarizing the research along with providing an outline of the limitations of this work and some suggestions for future work based on findings of the study.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter presents the summary of the in depth literature survey carried as part of this research. The literature review has mainly focused on cloud computing, quality of service in distributed system with special reference to cloud systems and trust and trust management systems. The main objective of the literature review was to understand cloud computing in detail with special reference to quality of service monitoring and how to quantize it. Special attention was paid to the current implementation of trust computing system in order to understand their design principles, strengths and weaknesses for incorporating them into the cloud systems for monitoring the service quality. The study highlights the drawbacks of existing mechanisms with a view of improving them.

2.2 Cloud Computing

Cloud computing has been considered the new paradigm in distributed computing as that has revolutionized the way hardware, software and other resources have been implemented, accessed, used and paid for [1]. Computing resources including hardware, operating system platforms, development tools and applications hosted in data centers are made available over the Internet for end users to access. Clients who access these resources would be required to pay only for the usage of the resources accessed. Prior to the introduction of cloud computing, computer systems especially servers were purchased in the form of hardware and hosted in client owned data centers or leased from public data centers. The cloud computing systems relieve customers from the burden of purchasing, installing and managing computer systems also reduces the cost of using these systems dramatically. Figure 1.1 shows the Capacity Utilization

developed by the Amazon Web Services [6]. From Figure 1.1, it can be seen that the demand for computing resources shown by the darker coloured erratic line are not uniform and shows variations in the short run as well as long run. Short term variations are mainly due to fluctuations on workload during a day and the week. Usually the demand for computing resources is high during business or office hours and goes down towards the end of the day and no demand at all during off hours. Similarly, there may not be any demand for computing during weekends depending on the type of work. Long term demand variations can occur due to seasonal patterns of work. The other important information that can be gathered from Figure 1.1 is that the average demand is showing a steady increase continuously. The increase in average demand is due to the growth of operations over time.

The commitment of computing resources is shown by darker coloured stepwise and lighter coloured erratic lines. The dark stepwise line shows the commitment of computing resources by traditional methods acquiring computing power, either through outright purchase or leasing from public data centers. The lighter coloured erratic line is the commitment of resources by cloud systems. It is very clear from the graphs that the traditional ways of acquiring computing resources do not follow the actual demand patterns and waste resources. If the capacity of the computing resources available is more than what is required, then the resources are underused; therefore wasting the precious capital invested. If the capacity available is less than demand, then customers would be frustrated making them leave for better services. Hence both overspending and underspending will result in loss to a business. On the hand, cloud computing commits resources exactly what is required, not less not more following the demand patterns both in short term as well as long term. The customers would be required to pay for only what is accessed or committed making the entire investment on computing immediately realized.

The cloud computing provides the right solution for any new business venture irrespective of type of the business. Usually, when a new business is started, the business pattern cannot be precisely predicted. Hence the amount that must be spent on computing resources may also not be known in advance. So, committing capital resources on computing may not yield the expected results. But, if the computing resources were purchased from public cloud services, the available computing power and other resources would follow the demand pattern closely making the investment fully worth. Also, using cloud computing makes the investment on computing an operational expenditure as opposed to a capital expenditure releasing the capital for the core activities of the business [4].

2.2.1 Cloud Computing Service Offerings

Cloud computing service offerings are currently made available in the market under three main categories namely, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [13]. IaaS is the raw virtual machine that has been created by the VMM installed on top of the physical hardware. Each created virtual machine presents itself to the user with its own CPU, memory, storage etc., depending on the requirement. This gives the impression that every customer has working on his own dedicated private system [52]. A virtual system based IaaS can be purchased as full system containing all the resources or individual components (such as virtual storage, virtual computing power etc.,) from different vendors and aggregate them together to form a single unit. PaaS is the offering of a complete development platform as a service on a pre-installed operating system [53]. Thus PaaS includes all the tools required for the development, testing and deployment of web applications such as programming languages, project management tools, application testing tools, application integration tools, Application Programming Interface (API) etc. SaaS is the offering of computer applications as a service over the Internet [54].

Under SaaS, the web based applications installed in the cloud specifically on top of both IaaS and PaaS can be accessed by customers as if they are accessing the locally installed software and pay only for the duration of access. In addition to these three main services, service providers have come up with innovative ideas and products that are hosted and delivered as services over the Internet [55]. Figure 2.1 shows these cloud service offerings in a structured diagrammatic format.

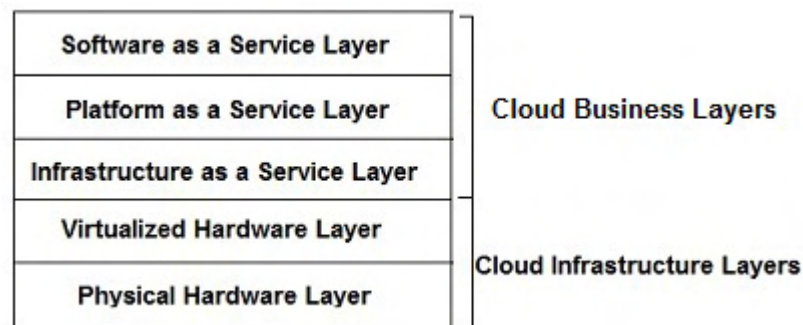


Figure 2.1. Cloud Computing Service Offerings

2.2.1.1 Infrastructure as a Service

Infrastructure as a Service or IaaS in short is the provision of virtual hardware as a service over the Internet to prospective customers [56]. Under IaaS, hardware resources including CPU cycles, storage space, memory, data, network etc., are made available on a virtualized system. The virtualization software installed and hosted on the real hardware creates multiple virtual copies of the hardware and present to the users as virtual systems. The virtualized systems thus marketed strictly resemble real systems and can be treated as real systems for all the practical purposes. Clients can install any operating system and application as if they have been installed on real systems. Hence more than one operating systems and applications can run on a single physical computer in parallel at any one time without interfering with each other. The VMM provides necessary security and isolation for these systems from interfering with each other.

2.2.1.2 Platform as a Service

Platform as a Service or PaaS is the provision of complete application development platform on a virtual system as a service over the Internet [57]. PaaS is installed on top of IaaS that provides the underlying infrastructure with the necessary access to physical hardware along with the security, privacy and isolation required. PaaS offerings usually include operating system, databases, development and testing tools, project management and coordination tools, deployment, monitoring, and management tools, API and infrastructure underneath them [58]. PaaS provides several advantages to the application developers compared to traditional development environments. These advantages include the elimination of the costs and efforts of evaluating, purchasing, installing, configuring, and managing all the hardware and software needed for enterprise applications, reachability of a larger customer base across a wider geographical area and lower cost for maintaining and delivering the applications over the Internet [58]. Due to all these advantages, the cloud market has seen several PaaS vendors hosting their services on the Internet in the recent times.

2.2.1.3 Software as a Service

Software as a Service (SaaS) is the model of hosting and delivering computer applications over the Internet on a utility basis. Web based applications that can be accessed using any standard web browser requires only slight customization to suit the individual requirements. SaaS eliminates the cumbersome operations of purchasing, installing and configuring of software, thus saving time and money for customers [47]. SaaS also provides the additional advantages of being accessible through the Internet from anywhere and anytime while eliminating the upfront costs and licensing costs. SaaS also lowers the in-house installed hardware costs as even low end computers can be used to access the cloud hosted applications. Cloud based SaaS applications are more reliable and scalable as their requirements are handled by the service provider

who enjoys the economies of scale due to larger customer base. SaaS applications also include many features in common including personalization and customization to suit the specific requirements of individuals. The capability to personalize the software provides the feeling that they have been exclusively accessed by users similar to the ones installed locally [59].

2.2.2 Cloud Computing Deployment Models

Cloud systems are divided into four main categories based on their deployment models. They are namely private clouds, public clouds, hybrid clouds and community clouds [60]. In private cloud systems, the complete computing infrastructure including the servers, networks and applications is implemented within the organization that owns and operates it. Public clouds are implemented by an independent commercial organization that sells its infrastructure to many organizations and individuals at a certain fee. A hybrid cloud is formed when an organization shares its computing demands between an in-house cloud system and a public cloud system. An organization may choose to implement any of these models depending on its requirements including economy of operations, security concerns, criticality of operations, efficiency or organizational policy. In addition to these three commercial cloud models, there is another model called community cloud model, where several organizations share the common computing infrastructure. Figure 2.2 shows the cloud computing deployment models in a graphical manner. The following subsections provide a detailed discussion on each of them.

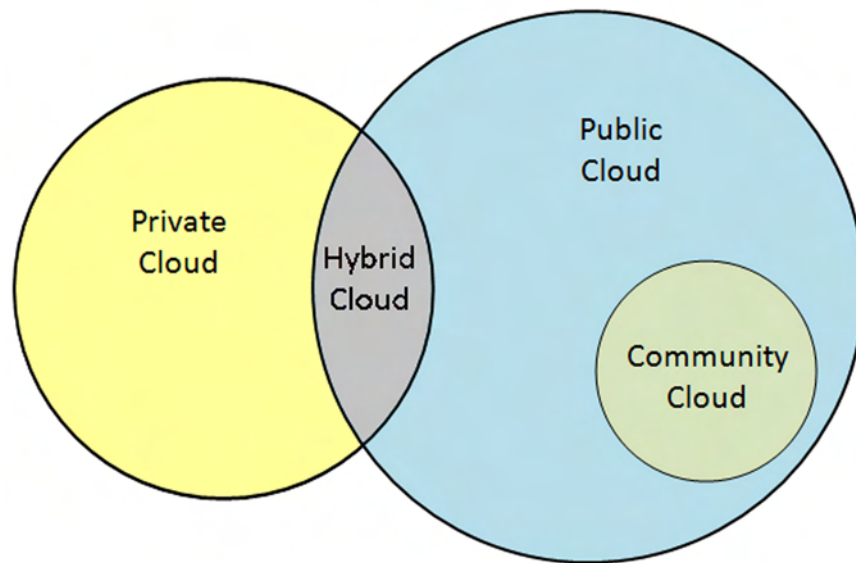


Figure 2.2. Cloud Computing Deployment Models

2.2.2.1 Private Cloud

When organization implements a cloud system including the servers, networks and software within the organization for its own use, it is called a private cloud system. They are also called internal clouds or enterprise clouds as the entire cloud infrastructure is housed behind the corporate firewall of the organization implementing it. The administration of such system is the responsibility of the organization and is carried out by the internal staff themselves. The clients of such a system are the various divisions, departments or units within the organization. Out of the three models, a private cloud is the most expensive one to implement while providing the highest security [61].

According to the Gartner Institute, a private cloud has five key elements, they are as given below:

- Both hardware and software resources are delivered as services.
- Possesses the flexibility to meet the requirements of the internal clients.

- Only shares the resources within the internal users of the organization.
- Has the ability to allocate the cost based on actual use.
- Standard Internet technologies including protocols, application and tools are used for resource delivery.

Organizations implement their own private cloud systems due to various reasons. Some of the common reasons for selecting a private cloud implementation and the advantages for the organization are given below:

- A private cloud can pool resources that are distributed across multiple departments and divisions while maintaining separation between them. Pooling resources help to consolidate resources at a single location enabling higher priority tasks to be allocated with higher capacities when required and allocating those resources to other lower priority tasks at other times.
- Ability to reduce the cost of computing by co-locating the entire computing infrastructure a single data center and manage it as a single unit.
- Freedom to define their own security as single unit and to manage it effectively.
- May act as starting point in embracing the cloud technology with the objective of moving to a private-public hybrid cloud on a later date.

2.2.2.2 Public Cloud

Similar to any other public infrastructure, public cloud systems are also setup, owned and operated by commercial service providers. The public service providers has the ability to draw benefits from economies of scale through the establishment of large data centers and distributing that cost among a large number of customers [62]. The business model adopted by the public cloud service providers is known as system either pay-as-you-go or utility model [1]. Due to the larger capacity of the public

cloud system resources compared to their private cloud counterparts, they have the capability of seamless on demand scalability for meeting customer demands. Also, the public cloud service providers can implement redundant geographically distributed data centers resulting in better robustness in times of disasters. One of the main factors to note when procuring services from a public service provider is that the same computing infrastructure with generic configuration, security protection, and availability variances are provided to all the customers.

Advantages

The advantages of public cloud systems over their private cloud counterparts are summarized below:

- Low capital investment: Generally no upfront implementation cost along with pay as you go payment scheme.
- Better development/testing environment for applications those scale to many servers.
- Improved robustness, scalability and resilience.
- Lower administrative cost.
- Better disaster mitigation capability.

Disadvantages

The public cloud systems have certain drawbacks too. They can be summarized as follows:

- Multi-tenancy and transfer of data over the public Internet creates a security threat.

- Reduced control over how the sensitive information is handled in a public data center.

2.2.2.3 Hybrid Cloud

A hybrid cloud system is created by merging a private cloud system with that of a public cloud system. In such a system, a portion of the workload is handled by the private cloud owned and operated by the organization while the balance is transferred to a public cloud over the Internet [2]. Generally, hybrid clouds are implemented for optimizing the advantages of cloud computing while minimizing the threats. Hybrid cloud systems enable the ability to draw the economic benefits of public clouds while flexibility of defining their own security plans for the sensitive data stored within the private cloud. Also the hybrid clouds improve the flexibility and scalability of computing through handling the base load within the organization while transferring the spikes to the external cloud system. Augmenting the private cloud with the resources of the public cloud would help successfully handle the unexpected demand surges.

The pros and cons of hybrid cloud systems can be listed as follows:

Benefits

- Flexibility: mission critical and secure applications and data can be run on private cloud while the public cloud can be used for development/testing purposes.
- Scalability: Peak and bursty workloads can be channeled to the public cloud while the private cloud handling the fixed base demand.

Risks

- Hybrid clouds are yet to be tested in a real world environment; hence hidden dangers may lurk there.
- Managing security in a public private shared interface may prove to be difficult: An adversary may enter the private cloud through the public portion of the cloud.

2.2.2.4 Community Cloud

A community cloud is created by sharing a cloud computing infrastructure by several organizations [60]. Generally, organizations sharing common requirements and concerns including security plans, compliance to standards, jurisdiction etc get together to establish a shared computing infrastructure based on cloud technology [63]. A community cloud system may be considered as a smaller version of a public cloud serving a limited clientele. The cost of operating a community cloud lies between that of public cloud and private cloud as it is shared by a selected set of users (organizations). A community cloud may be hosted in-house at a member organization or off-premises at a third party [60].

Similar to the cost, the pluses and minuses of a community cloud also lies between that of public and private cloud systems. The complexity of managing community clouds also more than that of private clouds as it is necessary to meet the demands of many customers (partners). Different organization may have their own requirements that conflicts with that of another partner organization. Hence there are several issues that need to be addressed before moving on to a community cloud system. Some of the prominent issues that need to be dealt with are summarized below.

- How to share the common expenditures such as support/maintenance and

operational costs, infrastructure/capital costs.

- How to manage the availability and service levels across the entire community cloud.
- How to manage the contractual and security implications of data spread across multiple organizations and in multiple domains.
- How to handle the legal impact of a service outage along with the responsibility for answering it.

2.3 Quality of Service in Cloud Computing

Cloud computing systems may host thousands of globally dispersed clients at any given time. These clients may access different types of services that have varying requirements depending on the type of clients, services and resources involved. In order to meet the requirements of clients and services, it is necessary to provide a certain level of service quality by the service providers. Nevertheless, providing a guaranteed service quality in such a challenging environment is not an easy task [64, 38]. Though it is a challenging task, several researchers have undertaken to develop mechanisms, frameworks and systems which could guarantee the service quality requirements of different services. These methods mainly concentrate on how to optimize the use of available resources while meeting the requirements of the customers. The following is a brief look at some of the mechanisms, frameworks and systems proposed in this regard.

The optimal resource allocation model for revenue maximization proposed by Feng et al. in [21] has been mathematically derived and tested using both synthetic and traced datasets. The proposed model performs better than heuristic optimization of resources in maximizing profits. But the application of this method is limited as it considers only the mean response time as the QoS attribute to be satisfied. For customers who

require guaranteed performance or at least a commitment in terms of a confidence level cannot be served through this model. Hence from the customers' point of view, the model has limited application and may serve only casual users. A framework for SLA management with special reference to managing QoS requirements has been in [38] by Buyya et al. This architecture successfully integrates the market based resource provisioning with virtualization technologies for flexible resource allocations to user applications. But the proposed architecture concentrates more on helping the service providers to optimize their resource allocation than looking at the service providers' performance from a customer's point of view. A standard QoS framework for managing cloud workflow systems has been proposed by Liu et al in [65]. The proposed framework covers all the four stages of cloud workflow namely, requirement specification, service selection, consistency monitoring and violation handling. The short-coming of this framework is that it does not specifically identify any QoS parameters and also does not discuss how to differentiate clients requiring different QoS levels. The workflow scheduling algorithm proposed by Chen and Zhang is based on Particle Swarm Optimization (PSO) [39]. The proposed mechanism can optimize up to seven parameters specified the users compared to traditional optimization techniques that consider only the workflow execution time. The downside of the proposed mechanism is that it lacks a monitoring scheme for catching QoS violations or punishing the violators. The cloud workflow scheduling model presented Li et al. is novel attempt in customizing the environment to suit individual requirements as it incorporates trust and QoS targets into the model [66]. In order to analyze the users requirements and design a customized schedule, the authors propose a two stage workflow model where the macro multi-workflow stage is based on trust and micro single workflow stage classifies workflows into time-sensitive and cost-sensitive based on QoS demands. The classification of workflows has been carried out using fuzzy clustering technique. The proposed model restricts the QoS parameters considered

to response time, bandwidth, storage, reliability and cost. Also the delivery of QoS is confined only to average values and no guarantee of service delivery is provided at least in terms of a predetermined confidence level. This is a strong limitation of the proposed technique as the users do not have the freedom to select their own QoS parameters and no guarantee of the QoS delivery at the least a statistical validation.

The set of heuristics for scheduling deadline-constrained applications in a hybrid cloud system in a cost effective manner in [40] attempts to maximize the use of local resources along with minimizing the use of external resources without compromising the QoS requirements of the applications. The optimization heuristics takes the cost of both computation and data transfer along with the estimated data transfer times. The main criteria in optimization is the maximization of cost saving. The effect of cost factors and workload patterns on the savings have been analyzed along with the sensitivity of the results to the different runtime estimates. The advantage of the proposed methodology is that it can select an optimized set of resources from both private (in-house) and public cloud systems for meeting the QoS requirements. But at the same time it suffers from certain weaknesses. Though it is concerned only about the deadline concerned applications, it does not consider the failures that may occur after the scheduling has been done. The failure will increase the cost of execution and affect the application in terms of quality. The scheduling heuristic proposed by Emeakaro et al. considered multiple SLA parameters in selecting the right environment before deploying applications in the Cloud [41]. The attributes considered include CPU time, network bandwidth and storage capacity for deploying applications. These parameters have limited application in real world systems as they need to be considered only during deployment. Once the applications have been ready for client access, the customers would be more interested in performance attributes including response time, processing time and availability. Hence this heuristic may

not have much practical significance in real world business environments.

Mushtaq et al. have investigated the effect of different factors on the Quality of Experience (QoE) of multimedia users in a cloud computing network [67]. The authors of this paper have grouped the factors that affect the QoE into four groups. They are namely network parameters, characteristics of videos, terminal characteristics and types of users' profiles. The data collected through different methods have been classified using machine learning techniques such as Naive Bayes, support vector machines, k-nearest neighbors, decision tree, random forest and neural networks. Out of these methods they have determined the best method for QoS/QoE correlation after evaluating them. Hence it can be concluded that this paper discusses more about the capabilities of machine learning techniques than about QoS or QoE. The QoS/QoE correlation is a case for evaluating the machine learning techniques. The main advantage of the lightweight framework for monitoring public clouds in [68] is its low loading on local resources. On the other hand, it does not monitor the QoS parameters such as response time, processing time etc., which the customers may be more interested in. Similarly, Zhu and Agrawal have presented a framework for handling adaptive applications in cloud systems in [69]. The proposed framework is based on multi-input multi-output feedback control model for resource allocation. But the model is limited to memory and CPU performance only, hence the application may be affected by the underperformance of other resources such as network, disk drives etc.

The profit model based on response time yields different results than that obtained using traditional metrics [70]. This model shows that both under as well as over allocation of resources affect profits. The right allocation of resources depends on many factors such as available resources, workload distribution, system configuration,

and profit model. This is an innovative method of analysis the effect of managing QoS on resource utilization. The results only discuss the effect of managing QoS, how to provide an optimal allocation is not discussed. The cloud resource pricing model proposed by Sharma et al. in [71] balances the QoS requirements and profits. This model uses the realistic valuation for underlying resources using the age of resources. The proposed model does not include utilization in computing the cost. Hence it may lead to inaccurate projections.

The distributed resource allocation algorithm proposed by Adami et al. can be used for both cloud and grid systems [72]. The algorithm is also capable of handling multiple resource requirements. The criteria for optimization applied in this algorithm is a compromise between the execution time and economic cost along with system and network performance parameters as additional factors. The proposed algorithm successfully incorporates many system and network performance parameters but fails to consider the failures that may arise after allocating resources. The failures arising after the allocation increase the cost of computation as they would require more time for execution. Hence the cost based optimization used in the proposed algorithm may not be accurate due this shortcoming. [73] reports the results of the performance of virtualized hardware of two IaaS providers. In this work, the Dwarf benchmarks for measuring the performance of these cloud providers has been used. This work shows that labeling the actual performance as small, medium or large does not actually reflect the true nature of a system along with the fact that some applications may run better on certain hardware than the other ones.

The extensible dynamic provisioning framework for multi tenant cloud system proposed by Gohad et al. enables efficient mapping of resources based on QoS requirements [74]. The proposed framework starts by defining a tenancy requirements

model for helping map provisioned resources. The other index called the health grading model handles the QoS characteristics of tenants. Together both these indexes permit dynamic reallocation for existing tenants depending on changing requirements or predicted health ratings. The proposed framework is innovative in the dynamic resource provisioning sense, but may not be suitable for applications that have bursty requirements. Also the proposed framework is based on starts small and grows large criterion. But when new tenants arrive, the allocated resources are not deallocated from the existing tenants, this would starve the new tenants of resources. The Nash and Raiffa have been combined in the bargaining solutions to arrive at an optimal allocation strategy in the optimum resource allocation strategy for cloud infrastructure based on bargaining [75]. The proposed strategy handles the dynamic nature of cloud very well during run time but the system does not permit to manage resources from multiple sources. Hence if a single service provider cannot meet all the requirements of the customer, he will be required to settle for a sub optimal allocation of resources. The expected future workload is an important parameter in optimum allocation of resources to different applications. In this direction, Sanchez et al. have investigated the capability of Markov arrival processes based queuing models to predict future workload of cloud systems [76]. The main downside of this model being tested only with numerical experiments. Hence the true capability of the model remains to be evaluated with real data traces.

An optimization framework for cross layer cloud services has been proposed in [77] by Kouki et al. The optimization across multiple layers has been carried out enforcing the SLA dependencies between them. The frame-work is very suitable for vendors marketing multitude of services and also takes the dynamic nature of cloud systems. The propose system currently lacks the run time management of QoS performance. Wu et al. in [78] have proposed some algorithms for resource allocation for SaaS

providers to balance the cost of hardware and SLA violations. This proposed algorithm takes certain QoS parameters such as response time and service initiation time for satisfying the customers while minimizing the use of hardware resources. These algorithms propose to reuse the already created VMs in order to minimize cost, but it may create security problems for customers as the residual information in the VMs can be used against them.

Chauhan et al. present a process model for identifying a cloud service provider for a given set of requirements by matching SLA parameters [79]. This process finds a match by creating two models called the capability model and requirements model, which are then translated to graphs for evaluating the compatibility. Based on the compatibility, each node pair is given a mark between 0 and 1 from which the final score is computed by summing them all. This is a good effort for automating the process of matching the customer's requirements with the service provider's capabilities. But, it does not consider the dynamic nature of the cloud services. It only matches the published capability of cloud providers with customers' requirements. This is a major shortcoming of this process.

From the above discussion, it can be seen that most of the mechanisms listed above focus on managing cloud workflows for enhancing the resource utilization, while maintaining the service quality. The main focus of this research is to monitor and measure the service quality of cloud service providers from the customers' perspective. The next subsection takes an in depth look at the mechanism proposed in the area of monitoring cloud services.

2.3.1 Service Quality Monitoring in Cloud Computing

For cloud computing to become the preferred choice of customers, they need to be satisfied with services they receive on many aspects [32]. Service quality is one of the main aspects that must be satisfied by cloud service providers [80]. As discussed in Chapter One, if the customers do not receive the service quality they were promised with, their entire business would suffer defeating the whole purpose of moving to the cloud. Also due to the attractiveness of the cloud services, many service providers have been offering their service to the prospective customers. This proliferation of cloud providers in the market has created a dilemma for customers to select the right service provider, who would meet their requirements including service quality [32]. Hence cloud service quality monitoring and quantification have become one of the most important requirements today.

Though the importance of cloud monitoring has been understood, it has not been given sufficient attention by researchers compared to other areas related to cloud computing [33, 37]. This research aims to fill this gap developing a set of mechanisms for effective cloud monitoring and quantification. In order to understand the current status of cloud monitoring, a thorough analysis of the available literature has been carried out. Following is the summary of the analysis carried out in this regard.

The importance of dynamically monitoring the QoS of virtualized services has been outlined in [42]. This work further claim that the monitoring of the services would help both the cloud providers and application developers to maximize the return on their investments in terms of managing the services and applications at peak efficiency, handling changes in service performance promptly, detecting SLA violations, failures of cloud services and other dynamic configuration changes. The researchers mainly concentrate on Simple Network Management Protocol (SNMP)

based QoS monitoring. Since this is a concept paper describing work in progress, no concrete proposal is put forward or evaluated. A framework for QoS Monitoring and Benchmarking for cloud computing is presented in a typical cloud Application As-a-Service in [29]. This framework mainly concentrates on monitoring application components rather than the service quality of the underlying infrastructure. Hence the proposed cross-layer detection mechanism is very narrow in scope and cannot be adapted to monitor IaaS services. On the other hand, the monitoring mechanism for storage clouds in [81] could monitor only storage clouds and lacks the capability for generic implementations. Thus this mechanism falls short of the requirement of monitoring the cloud systems for the purpose of ranking any cloud system. On the extreme side, the client application for monitoring cloud QoS for iOS5 proposed in [82] can be used by clients to monitor the performance of their cloud provider. The main shortcoming of this application has been its limitation applicability for general implementation as its design focuses narrowly on the available transfer rate and one-way delay.

The QoS architecture based framework for ranking of cloud services presents an extensive set of QoS parameters along with simple mathematical formulas to compute them [83]. In this framework, the cloud service ranking unit has been treated as black box and no implementation detail is given. This paper also falls within the category of concept papers rather than one that is presenting concrete mechanisms or implementation. On the same lines, the framework for ranking cloud services proposed in [32] also defines the common service quality attributes providing mathematical equations . The paper also introduces a weighting scheme for the different parameters based on the user preferences. But the proposed method uses static values for computing the ranks and unable to continuously track the performance of the cloud system. Similarly, the QoS based cloud service provider selection

framework proposed in [84] contains a module with a ranked list of cloud service providers. But there is no information about how this ranking has been carried out nor any mechanism presented. Hence this work can only be considered as a conceptual idea identifying the importance of cloud monitoring from the customers' perspective.

Deviating from all the above work, some researchers have proposed the novel idea of bringing trust computing into monitoring QoS of cloud systems, which have been hitherto treated as two different domains. The trust management system for IaaS providers proposed in [48] presents the idea computing trust using service quality parameters. Though this is a novel proposal, the paper stops short of proposing any mechanism. Thus, this is only a concept proposed in the direction of combining the service quality and trust. The work proposed in [49] puts forward a method for aggregating multiple service quality parameter values into a single trust score. It then explains how this single score can be used to rank different service providers. The mechanism proposed is simple and can be implemented by an independent cloud monitoring service, so that prospective customers could consult this monitoring service for identifying a suitable service provider. The main shortcoming of the proposed mechanism is the non consideration of the dynamic nature of cloud computing. Also the proposed mechanism is a standalone one that can monitor only a small set of cloud providers in a specific region only. Hence it would not be feasible for such a system to identify a suitable provider, even if he is providing acceptable services, if he is not within the reach of the monitor. On similar lines, the trust computing and management model proposed in [85] is claimed to have used multiple QoS attribute to compute the trust value, but there is no clear explanation how these parameters are combined or how to prioritize one parameter over other ones depending on the user requirements.

Table 2.1 summarizes the service quality monitoring mechanisms/ models discussed above highlighting the strengths and weaknesses of each of them.

Table 2.1

Summary of Features of the Service Quality Monitoring Mechanisms

Work	Model	Comments
[42]	SNMP-based QoS monitoring	Only a concept paper and no concrete mechanisms for implementation is presented. Generally SNMP information is not available to 3 rd party monitors or clients. Hence lacks the capability of being implemented as an independently deployed monitoring tool.
[29]	QoS monitoring and Benchmarking as a Service	Concentrates solely on monitoring application components. Cannot monitor the performance of underlying infrastructure.
[81]	Monitoring mechanism for storage cloud	Can monitor only storage systems. Lacks the capability for monitoring generic cloud services.
[82]	Client application for monitoring cloud QoS	Implemented on iOS5. Very narrow implementation focus with only two parameters; available transfer rate and one-way delay.
[83]	QoS architecture based framework for ranking cloud services	An extensive set of QoS parameters along with their mathematical formulas. But the proposed framework has been treated as a black box, no implementation details are provided.
[32]	Framework for ranking cloud services	The framework introduces a weighting scheme for parameters. The proposed framework lacks the capability to continuously track the performance of cloud systems.

Table 2.1 continued

[84]	QoS based cloud service provider selection framework	Contains a ranked list of service providers. But lacks in detail how this ranking has been arrived at. Also no mechanism has been presented for preparing such a ranking.
[48]	Trust management system for IaaS providers	Proposes the novel trust computing mechanism using service quality as input. No mechanism is presented only an idea was proposed.
[49]	Trust model based on QoS	Introduces the notion of single score for ranking different cloud services. The proposed mechanism does not take the dynamic nature cloud system performance into account.
[85]	Trust computing and management model	Claims to have used multiple QoS attribute to compute the trust value. No clear explanation how these parameters are combined is provided.

From the above discussion, it can be observed that the mechanisms and frameworks proposed in the literature suffer from several shortcomings. For example, the work of Alhamazani et al. try the capture the QoS from the existing network services such as SNMP for capturing the performance parameters of cloud services, the work of Stoicuta et al. is too narrow to employ in a general setting and the work of Goyal et al. is a good attempt at quantifying the QoS performance of cloud services but it does not allow users to select the required parameters or prioritize them. This is a major shortcoming as QoS demands of cloud services are broad and different from one service to another. Though there are shortcomings, Manuel's work is in the right direction as it can be implemented by an independent monitor. This mechanism can be further developed for a more complete one, if the dynamic nature of the cloud system can be incorporated into it. In order to handle the scholastic nature of cloud service

quality parameter values, the researcher proposes to incorporate probability into it as an enhancement. Also the enhanced mechanism must be capable of monitoring large number of cloud systems that is distributed across a large geographical area.

2.3.2 Service Quality Parameters in Cloud Computing

Prior to developing any mechanism for the quantification of the service quality for ranking service providers, it is necessary to identify the parameters, which can differentiate the services in an objective manner. Cloud computing is the result of successful integration of many technologies such as virtualization, web 2.0, service oriented architecture, web services, utility computing, etc [86, 87, 88]. Thus the performance of cloud computing depends on the performance of these individual components as well as the integration between them. Hence the parameters to be identified will come from many technologies but with the same focus on identifying the performance quality.

Service quality can be categorized into two specific groups, such as service performance in terms of customer expectations and service providers' targets [89]. Customers expect better services at the lowest possible cost while the service providers want to maximize their profits through optimum utilization of their resources. The main objective of this research is to quantify the quality of service of cloud providers from the customers' point of view, so that the best service provider meeting their requirements can be identified.

A high level set of parameters representing the business-relevant Key Performance Indicators (KPI) for establishing a standard method for measuring and comparing business services have been presented in [32]. These parameters include accountability, agility, cost, performance, assurance, security and privacy and

usability. Within these broad set of parameters they have identified a further subset of factors that make these broad parameters. Though this is a good effort identifying the factors that would make cloud computing more acceptable to customers in terms of providing them right information on the service providers, it fails to give specific definitions and mathematical formulations of the metrics at narrowest levels. Hence, it is necessary to consult studies on the quality of service of different but related technologies that make cloud computing possible. On the same lines, Xiong and Perros state that the QoS can be defined based on parameters such as response time, network utilization and throughput In [27]. This work concentrates only on modeling QoS using percentile response time. The main reason cited for considering only the response time is that in their own opinion, response time plays a much more important role in the user satisfaction than all the other QoS parameters. The same argument is placed by Bochmann et al. as well in favor of response time for electronic commerce applications [90].

Multiple QoS parameters contribute to the satisfactory performance of many Internet based applications [91]. It has been noted that transactional applications such as web services demand better response times and throughput guarantees while non-interactive batch job are more concerned with performance in terms of job completion time and accuracy. The same conclusions have been arrived at in [92] as well. It further states that interactive applications, due to their inherent nature of having an overall short duration, lend themselves to automation at short control cycles. On the other hand non-interactive jobs generally require calculation of a schedule for an extended period of time [92].

A comprehensive set of parameters that can be used for web service recommendations has been listed in [93]. These parameters have been categorized into multiple groups

such as runtime related, transaction support related, configuration management & cost related and security related QoS. The detailed set of parameters thus identified include, Runtime related QoS: scalability, capacity, performance (response time, latency, and throughput), reliability, availability, robustness/flexibility, exception handling and accuracy; Transaction support related QoS: transaction integrity; Configuration Management & Cost related QoS: regulatory, support standard, stability/change cycle, cost and completeness; Security related QoS: authentication, authorization, confidentiality, accountability, traceability and auditability, data encryption and non-repudiation.

The factors that change dynamically are considered to be more important than the static ones in determining the overall quality of cloud services. This is mainly due to the reason that clients choose cloud computing for their capability of dynamically adjusting the resource allocations [94, 95]. The dynamic behavior of distributed systems such as cloud computing not only depends on the capacity of the system but also the current loading of the systems as well [96]. Hence the performance of the system may degrade even if the amount of capacity of the system is large but also need to serve a large number of customers concurrently. Also the random arrival of user requests at the cloud data center may make certain servers overloaded while others idle, if the systems are not properly managed [96]. Hence from the customers' point of view, dynamic performance of the systems play a greater role in meeting their performance requirements than the static ones such as cost, regulatory compliance, customer support etc.

Hence only a selected set of dynamic performance parameters has been selected for further analysis in this research. The selected parameters include response time, service time or job completion time, availability, reliability and integrity. Though

only a specific set of parameters have been considered in this work.

2.3.3 Definition of Performance Metrics

The definitions of the performance metrics identified in the previous subsection are given below:

Response time: is defined as the total amount of time elapsed between a request made by a client and the initial response received from the server [97]. Response time is considered as one of the important parameters from the customers' perspectives [90]. Generally the response time of a distributed system is a result of multiple delays such as server delay, network delay, transmission delay, queuing delay at various points etc. in cloud computing there are additional delays such as identifying the necessary resources for launching a virtual system and the actual time taken for spawning the virtual server adds to all these delays. Out of all these delays, the time taken for launching the virtual servers has major impact on the performance of cloud systems [98]. The response time of public cloud service providers are not constant and show large variations even among large and established service providers [99]. Hence response time is one of the important parameters for measuring the performance or the service quality of the cloud computing service providers.

Service time or Job completion time: also known as processing time is the total time taken for a process to complete from the time of submission to the time of delivery of the completed results [100, 101]. Service time depends on many factors such as number of subtasks in the request, whether these tasks can be executed in parallel or need to be carried out in sequence etc. It also depends on the system parameters such as speed of the CPU and/or percentage of CPU allocation and sometimes speed and amount of memory available too [102]. Service time plays a major role in determining

the performance of a system as this is the period how long an application occupies the system resources. When a service request is placed with a cloud system, that request can either be successfully completed or fail due to some reason. If the request fails to complete in the first attempt, that request can be reattempted using one of the four fault tolerant techniques [103]. Irrespective of how the task is completed, once results have been successfully delivered, it is considered a successful completion of a request. But the total processing time will be different depending on how many times the task was attempted or the type of fault-tolerance employed by the system. This would affect the user perception of service quality as he has to wait for a long time to get the final results. Hence service time is an important parameter in the performance of cloud computing.

Availability: is the proportion of times, a system is up and ready to accept and process jobs from clients immediately [104]. A system may become unavailable due to various reasons such as systems failures, power outages, network availability, too busy to serve or malicious attacks such as denial of service. Businesses heavily depend on the computer systems for their profitability. They lose business as well as customers during the non performance of their systems [105]. One of the main reasons for businesses shifting their computing requirements to cloud is the expectation of better availability and responsiveness [106]. Thus availability becomes an important factor in deciding the service quality of a cloud systems provider.

Reliability: is the percentage of successfully completed jobs in a given time [107]. Public cloud service providers manage large data centers with hundreds of thousands of servers equipped with various kinds of hardware resources. Irrespective of type of technology used for manufacturing these hardware components, they may fail sometimes within their specified lifetime. The probability of failure increases with

the size of the cloud computing system as the number of components making such a system also becomes large. A single failure may affect the operation of an entire active application disrupting many users [108]. Since failures affect the customers directly, they will have an adverse effect on the business performance. Hence cloud computing reliability is an important factor of service quality that requires special attention of the providers.

Integrity: is the assurance provided by the service provider that the job assigned will be carried as expected and the results produced are authentic and reliable [109]. Integrity of the operations can be achieved through ensuring that the request is not tampered with prior to or during execution by a rogue user or malicious software agent [110]. Also it is necessary to ensure that the work is carried out in reliable and trustworthy computers as malfunctions may also modify the information. Similarly data integrity is also an important aspect in cloud computing as it needs to be protected from unauthorized access and modifications [111]. Thus integrity is also an important factor in the service providers' service quality.

2.4 Trust Computing and Management

Since it has been proposed to monitor the cloud service quality and compute a single quantity similar to a trust score, it would be pertinent to look into trust and trust computing at some depth. Pioneering work in trust and reputation has been carried out by the researchers in social sciences for understanding the character and conduct of societies [112]. Trust is a topic that has been extensively studied by researchers in many fields including psychology, political science, sociology, anthropology and economics [113]. The main focus of the psychologists studying trust is the mental attitude. They study the workings and behaviors of human mind during the times when he trusts or distrusts someone [114]. Several cognitive trust models have been

developed based on this notion [115, 116, 117, 118]. Social relationship between people has been the sociologists approach to trust. During social interactions people develop various degrees of trust between each other and the trust thus developed play a major role in their future dealings and outcomes. In the field of computing, social context of trust has been exploited for successful implementation of multi agent systems, e-commerce, social networks and other interactive systems [113, 119, 120]. The similarity between human social interactions and computer based systems include how they collect, filter and disseminate data for creating useful information to suit the situations. The value of trust depends on its utility for economists [121]. Computer scientists commonly use Game theory a tool that studies the interactions involving conflicts and cooperation between rational minds with the goal of increasing the individual values. The different strategies adopted during these interactions create different levels of trust based on the outcomes of the strategies and actions [122, 123]. Prisoner's dilemma, diner's dilemma, platonia's dilemma and the tragedy of the commons are commonly used for studying how and why different individuals behave in a specific manner [124, 125].

All these studies have contributed positively to the field of computer science as they help understand the workings of human mind the resulting behaviors of people at different occasions

Researchers in computing and related fields could exploit the benefit of these studies as they provide a critical perception into human behavior under different circumstances [119, 126, 127]. Open and distributed systems including peer to peer networks, client server networks, grid computing, cluster computing, semantic web, ontological modeling, web services, mobile ad-hoc networks and e-commerce have all employed trust and reputation based algorithms, mechanisms and techniques for

various operations [128, 129, 130, 131].

The diverse and rich literature available on trust and reputation has created a mixed situation with both advantages and drawbacks. The main advantage of the availability of the literature from multiple fields for the computer scientists is that it provides a deep insight into the field so that these models can be readily implemented. On the other hand, diverse notions of scientists from different fields on a single subject create confusion due to non-agreement for a single definition of trust. So, presently trust means different things for different people and they use terms like attitude, belief, probability, expectation, honesty and so on to suit their requirement and situation.

Though there is no agreement among scientists from different domains on a common definition of trust, some common key factors and attributes can be identified in almost all the definitions. They are;

- Trust becomes important only during the uncertain and risky situations.
- Trust becomes the basis for many decisions made during critical situations.
- Prior knowledge and experience plays a major role in creating trust or distrust.
- Trust is subjective and may depend on an individual's opinions and values.
- Trust is dynamic as recent experience and knowledge resulting from it has the ability to dominate over the old experiences and knowledge.
- Trust is circumstantial.
- Trust has more than one dimension.

According to McKnight and Chervany, trust is an aggregation of 16 characteristics which can be grouped into five categories [114]. The main categories and characteristics under them are as follows:

Competence:	capable, expert, active.
Predictability:	certain.
Benevolence:	moral, goodwill, caring, responsive.
Integrity:	honest, credible, reliable, dependable.
Others:	assailable, safe, shared agreement, personally appealing.

Trust relations can be grouped into three main categories as hierarchical trust, social groups and social networks [132]. Hierarchical trust creates a tree structure of relationships with nodes representing individuals and edges symbolizing the degree of trust between them. This tree structure allows any two specific nodes to specify a trust degree between them by transiting through intermediary nodes. Social groups are bound together through common well identified goals and objectives. The members of a group share common interest information by propagating them to all the other members. Trust plays an important role in creating and maintaining the cohesiveness of social groups. Social groups are represented using graphs, where nodes are members and the edges are links between them. Social networks are created by establishing relationships with other individuals in a community. The word relationship in this context has been given a special meaning as to the interaction between the individuals in a network depending on the mutual understandings. By walking through a social network, it is possible to discover the relationship paths between the individuals in a network.

The four types of trust identified in [133] by Zhang et al. are as follows:

- Subjective vs. objective
- Experience-based vs. opinion-based
- Global information-based vs. localized information-based

- Ranked vs. threshold-based

Objective trust is created when an entity's trustworthiness is measured objectively against a universal standard. On the other hand, if the trust measured depends on an individual's preferences and interests, it is known as subjective trust. Decisions made based on the results of an individual's interactions are known as transaction based trust and the trust built from the mere opinions is the opinion based trust. If information from each and every node in a system is collected before building trust, then it is known as either global trust or complete trust. On the other hand, if the information is collected only from one's neighbors, it is called the localized information trust. When the trust worthiness of an entity is ranked from the best to worst or vice versa, it is rank based trust whereas if the trust is declared yes or no depending on a preset threshold value it is known as threshold based trust.

2.4.1 Trust Management in Cloud Computing

The popularity of cloud computing and the importance of trust management in distributed systems have attracted attention of researchers from both academia and industry. They are actively involved in developing models, methods, mechanisms and techniques to make cloud computing trustworthy and acceptable to majority of users. In this section, special attention has been paid for scrutinizing and understanding the recent developments in management of trust in cloud computing and organizing them in a coherent way for easy reference. It has been identified that it is possible to enhance the security and dependability of cloud systems by integrating a trust computing module into the existing cloud systems [134, 135]. This conclusion has been arrived at after an in depth analysis carried out by the authors on the security of cloud computing environments. It has also been proposed that the cloud trust module must be entrusted with ensuring the confidentiality and integrity in cloud

system through proper authentication techniques.

In order to understand trust management in cloud computing from a user's point of view, what a cloud user wants from their service providers with respect to data security and privacy has been examined in [136]. For the purpose of answering the issues raised by prospective customers, the strategies that can be adopted by service providers for enhancing the trust of customers on service providers has also been highlighted in this work. The main aspects that play a vital role on the level of trust customers place on cloud services and service providers include control, ownership, and security. The main issues identified by the researchers for reducing the trust of customers in terms of security and privacy of data include diminished control over the data and lack of transparency from the service providers side. The researchers of this work further state that certain actions from the cloud service providers' side may act positively to enhance the trust of customers. The identified actions that could enhance the user level trust on service providers include the provision of remote access control facilities to personally customize the security of user resources, automatic tracing and auditing features to enhance the transparency of service provider actions, independent third party certification of cloud security properties and capabilities and the provision of a security enclave where users can define their own policies and mechanisms that govern the security of their resources. A combined security framework integrating different modules for handling trust and security related matters has been proposed in [137]. The main issues addressed by this framework include identity management, access control, policy integration between different clouds, trust management between service providers and customers and trust management between different service providers. The three main stakeholders specifically identified this framework are customers, service providers and service integrators. The service integrator acts as the mediator bringing the service providers and customers together. In order to achieve his

main goal, the service integrator discovers the services provided by various providers, negotiates with them and composes different service bundles by combining services from collaborating providers with the objective of meeting customer requirements. The trust between different parties in the system is also managed by the service integrator. There are four main components in the service integrator for managing security, trust, service and heterogeneity. Each unit is assigned with specific tasks to handle especially the heterogeneity management module handles the diversity of different service providers. Further to these, there are other minor units for handling minor but important tasks. Though it is a comprehensive proposal, it lacks any proof of concept in the form of a prototype implementation. In a similar work, Song et al. has found that providing unrestricted access by a remotely installed application to private data in SaaS creates several security related issues [138]. A mechanism that can independently verify the security by separating the data from the application while creating a trusted binding between them has been proposed for addressing these issues. The proposed mechanism anticipates the collaboration of four independent stakeholders for achieving the maximum security. They are namely, the providers of hardware resources, the application and the data along with the coordinator. Generally the providers of application and data are the owners of the respective resources, the resource provider is responsible for the provision of ancillary services such as searching and location of resources and the implementing a secure interface for data processing using the application. The coordinator plays the role of a broker bringing all the other parties together.

The proposed mechanism suggests the following steps for successful implementation of security:

- The application and the data to be uploaded by the respective owners to an

independent resource provider. It is necessary to encrypt these resources prior to uploading them to an accountability vault module provided by the resource provider.

- With the aid of a coordinator, the data provider finds a suitable application and executes it on his data.
- An execution identification tag is provided to the data provider when the application begins its operation.
- Once the execution is completed, the results are dispatched to the data provider's interface. The results thus delivered can be displayed on screen, printed in a hard copy format or downloaded as it is by the data owner.

The service charges paid by the data provider will be shared by the application providers of resources (hardware and application) and the coordinator based on their contributions. On the completion of the execution, an operation log is generated and posted on the application provider's interface giving the type and duration of use of the application without disclosing the identity of the data provider and the content for security purposes. This operation log helps the application provider to keep track of the utilization of his application. This proposed mechanism suffers from certain apparent shortcomings in terms of its claims. There is no guarantee that the application will not make its own copy of the data processed. Usually application providers furnish only a brief description of their software along with the algorithms used. No source code or implementation details are made available to third parties outside the development team for protecting the intellectual property rights. Hence it is not possible to guarantee that the software is free from any malicious code. Also the application enjoys complete control over data as it runs with data owner's rights and privileges. This kind of security threat cannot be prevented even with a detailed audit trail as they are unable to detect them.

A cloud trust model based on social security for managing security and related issues has been presented in [139]. This model classifies the specific cloud security issues identified social insecurity problem. A three pronged approach has been proposed to address the security issues by dividing the social insecurity problem into three sub problems. The three sub problems thus identified are problem of multiple stakeholders, problem of open space security, and problems associated with handling of mission critical data. The problem of multiple stakeholders in cloud computing has been identified to arise due to the interaction of many stakeholders including customers, service providers and intermediaries. At the beginning of the contract, the customer and the service provider sign a service level agreement. Based on the conditions stipulated in the agreement, the customer transfers some of the some of the powers or authorities on his resources to the service provider. Though the customer may like have the same security policies applied to the cloud hosted resources that he would apply on the in-house hosted resources, there may be differences due to providers' limitations. A provider is required to serve multiple customers, hence he is unable change his policies to meet the specific demands of a single customer. Also, only the conditions stipulated in the SLA binds the service provider to the customer, hence the policies controlling the resources may be more relaxed than what a customer would like to have. The SLA plays an important role in these situations by acting as a glue between the service provider's delivery and the customer's expectations. In the authors' opinion, the data hosted on the cloud becomes open and accessible to anybody including third parties depending on the strength of the authentication process of the service provider.

The problem of open space security arises due to the delegation of control over the data to the service provider. The service provider practically decides where and how the data is physically stored and managed. The advice of the authors in this regard

is to have the data encrypted prior to transferring over the Internet. Once the data has been encrypted, the security issue takes a new turn from the data security to the security of the cryptographic keys. The cryptographic keys that have been employed in encryption and decryption must now be protected without compromising the three pillars of security, confidentiality, integrity and availability of the data. The mission critical data handling problem concentrates on the issues that may arise when the administration of mission critical information has been transferred to a third party. The solution proposed by the authors for handling this kind of issue is to maintain a hybrid cloud computing system where only the less important data is hosted in the public cloud while retaining the full control over the mission critical data in private portion of the system. However this proposal may not be scalable to meet the requirements of small and medium enterprises as setting up a private cloud may incur high costs. The only way to overcome this issue is to enhance the security of the public cloud system. In order to address the issues discussed above, a mechanism known as cloud trust model has been proposed in [139]. This model comprises of two additional layers called internal trust layer and contracted trust layer in addition to the conventional trust architecture. The internal trust layer installed within the customer premises on systems owned and managed locally. The internal trust layer is the foundation on which the entire trust architecture is built upon. The main function of the internal trust layer is managing credentials and security keys. The mission critical data requiring extra security is also stored under this layer. Contracted trust is enforced through an agreement namely the service level agreement. The service provider affords trust to the customer through the agreement signed between them. The contract usually contains three main documents namely, the Service Policy/Service Practice Statement (SP/SPS), the Identity Policy/Identity Practice Statement (IdP/IdPS) and the service contract. The degree of trust preferred is negotiated by the parties based on the security needs of the information. A cloud computing system protected through all

these policies and mechanisms is identified as a secure cloud system by the authors. The domain-based trust model proposed in [140] ensures security and interoperability of cloud and cross-cloud environments. The domain-based trust model incorporates a security framework along with an independent trust management module built on top of traditional security modules. In addition to the trust model, a trust based security strategy for the safety of both customers and service providers has also been proposed in the same work.

A fundamentally different trust model based on family gene technology has been proposed in [141, 142]. This model deviates to a great extent from the traditional trust models that rely on public key infrastructure for handling the authentication and authorization processes. An in depth study on the basic operations of Authentication, Authorization, Accounting and Auditing (AAAA) management processes along with access control has been carried out and based on the findings, the authors have proposed the Family-gene Based model for Cloud Trust (FBCT) that integrates all these processes.

CARE resource broker is a framework for metascheduling of virtual grid resources in an efficient manner [143]. This CARE resource broker has been integrated with Kerberos and PERMIS (PrivilEge and Role Management Infrastructure Standard) authorization standards for creating an enhanced resource broker in [144]. In this enhanced trust model, the resource broker itself evaluates the trustworthiness of the resources prior to selecting them. The proposed system can be used for computing trust scores of both grid and cloud computing resources. The proposed model consists of three primary units named security degree evaluation unit, feedback evaluation unit and reputation trust evaluation unit for computing trust. The security degree evaluation unit carries out an assessment of resources depending on authentication/authorization

mechanisms and self security capability mechanisms. The model has been designed in such a manner that it supports different authentication and authorization mechanisms along with self security assurance capabilities. The trust score thus computed depends on the strength of individual mechanisms employed. The feedback evaluation unit also has three distinct sub units known as feedback collection, feedback verification and feedback updating units for aiding in the computation of its trust scores. The reputation trust evaluator computes its own trust score based on the capabilities of the resources accessed in terms of computational and network parameters. The final trust scores is the arithmetic sum of all three individual scores computed.

The SLA based trust model proposed in [145] consists of many components including SLA agents, consumer management module and services catalog. The SLA agents are the primary units playing the main role in this model by grouping the customers to classes depending on the needs, designing QoS parameters, negotiating with and selecting the cloud providers based on the capabilities and non functional QoS requirements and monitoring the activities on the customers' behalf based on the agreed upon QoS parameters. The main function of the consumer module is soliciting and executing certain services outside the cloud system. Cloud service directory is the common platform for providers and customers. Service providers advertise their services on the directory giving all the details including type of services and capabilities. Customers search the same directory for identifying the right service provider who could meet their demands in terms of functional and non functional requirements. The proposed model has not been implemented or tested for its functionality and effectiveness. A fuzzy theory based trust model for cloud computing is proposed in [50]. This trust mechanism considers response time as a basis for computing trust. Thus, this mechanism lacks the support for including multiple service quality parameters in its computation. The strength of this mechanism lies on the fact

that it is based on strong theoretical foundation and has the capability to continuously evolve the trust score based on performance.

The multi-tenancy trusted computing environment model (MTCEM) a two level transitive trust computing mechanism delivers trusted IaaS services to customers [146]. MTCEM also possesses the additional capability of security duty separation between stakeholders. Cloud systems in the public domain include multiple stakeholders including service providers, customers and intermediaries. Hence the resources may at the same time be part of multiple security domains serving different security specific functions. The objectives of the different stakeholder may be different and demand radically opposite outcomes such as the best services by customers and the maximum return on investments by the service provider. Both these demands cannot be met at the same time without compromises. This requires the cloud system to be capable of compartmentalizing the customers and service providers for the purpose of separation of powers by clearly defining responsibility boundaries for both parties. MTCEM adopts a two-level hierarchical transitive trust chain model supporting separation of duty. MTCEM identifies and supports three distinct types of stakeholders with specific functions and duties. They are service providers, customers and auditors by name. The service provider ensures that the infrastructure maintained by him is trustworthy while the customer responsibility begins at the guest operating system installed on the service provider's infrastructure. The duty of the auditor is to monitor the services of the providers on customers' behalf in order to ensure the conditions stipulated in the service agreements are met. The prototype system has been implemented as a proof of concept but lacks any performance evaluation.

The existence of firewalls in networks has been overlooked in many of the trust models proposed in the literature [147]. This conclusion has been arrived at after

a comprehensive study of existing trust models and firewall technology. This has been identified as a major shortcoming as firewalls are indispensable elements in any corporate IT infrastructure. In order to overcome this shortcoming, a firewall-through trust model based on cloud theory has been proposed in the same work. This model proposes a dynamic trust computing mechanism that continuously updates the scores based on the performance. Though this model has certain advantages, it has not yet been implemented or tested. The advantages of this model over other ones include:

- It implements different security policies based on stakeholder domains and requirements.
- The final trust score is computed considering the transaction context and the historical data arriving at a value reflecting both.
- The trust model is consistent with local firewall control policies requiring no changes.

A watermark-aware trusted application execution environment has the ability to overcome the security issues in cloud computing [148]. Such a watermark-aware environment can protect the applications from undeclared dangers hidden in the cloud systems. Two main components namely, the administrative center and the cloud server environment in the proposed model clearly separate the responsibilities and the functions of the parties involved. The main function of the administrative center is to embed a watermark for creating a customized Java Virtual Machine (JVM). The collection of customized JVMs creates a trusted server system, where only complete and specific Java programs are allowed. All the other non validated programs are barred effectively eliminating all the malicious invasion applications

An identity management of cloud systems independent of any third party involvement

has been proposed in [149]. The proposed system uses predicates over encrypted data enabling multi-party computing with both trusted and untrusted hosts in cloud system. The proposed system is rugged in the face of correlation and side channel attacks due to non involvement of third parties in the authentication process. But the system can be affected by denial of service attacks due to the non execution of active bundle in the remote host.

Table 2.2 presents the summary of all the trust computing mechanisms discussed above.

Table 2.2

Summary of Trust Computing Mechanisms for Cloud Computing

Work	Mechanism/Model	Comments
[136]	(No model is proposed in this work)	This is only a discussion on how to create trust on service providers from the users' perspective.
[138]	Trusted binding between data and application	It discusses how to isolate the data from applications and to create a trusted binding between them. Details of the implementation is not provided.
[139]	Cloud trust model for managing security related issues	The model is based on social security. A three pronged approach has been proposed to handle the security related issues. Concentrates only on security, no other issues including performance has been discussed.
[50]	Fuzzy theory based trust model for cloud computing	It supports only a single parameter namely response time for computing trust. It lacks the capability for incorporating multiple parameters.

Table 2.2 continued

[140]	Domain-based trust model	Ensures security and interoperability of cloud systems. Mainly focuses on security. No discussion on quantifying user trust has been carried out.
[141, 142]	Trust model based on family gene technology	This model concentrates on integrating the AAAA process rather than quantifying the user trust on cloud systems.
[143]	CARE based trust computing mechanism	This mechanism integrates CARE resource broker with Kerberos and PERMIS for creating an enhanced resource broker. This mechanism also concentrates on authentication and authorization, rather than on quantifying user trust based on the performance of the system.
[134, 135]	Trust module for enhancing security and dependability	A cloud middleware named Trusted Platform Software Stack has been developed for enhancing two pillars of security, namely confidentiality and integrity. This attempt also concentrates on enhancing security of the cloud system rather than measuring the dependability of the system from the customers' perspective.
[137]	Security framework for handling security and trust	The issues handled by the framework include identity management, access control and policy integration between two cloud systems. These issues mainly concern the service providers. Hence this framework is also not suitable for the identifying the performance of service providers from the customers' perspectives

Table 2.2 continued

[145]	SLA based trust model	Discusses mainly the importance of SLA in matching the customers' requirements to providers capabilities. Though it has several important modules such as SLA agents, consumer management module and services catalog, it is not possible to evaluate this model as this system has not been implemented. Hence this proposal also falls under the category of conceptual ideas towards coming up with a trust computing mechanism for cloud computing.
[146]	MTCEM	This is two level transitive trust computing mechanism with the ability of security duty separation. Though this model is proposed to have many positive things, it has not been properly implemented and evaluated. Hence this is still a work in progress.
[147]	Firewall-through trust model	Intends to incorporate the existence of firewalls in to the trust model. Contains a dynamic trust computing module for updating trust scores continuously. This model is also yet to be implemented or tested.
[148]	Watermark-aware application environment trusted extension	This mechanism also concentrates on security of cloud systems rather than performance. It tries to protect the application from dangers. Hence this mechanism is suitable for service providers rather than helping the customers to identify the suitable cloud provider.

Table 2.2 continued

[149]	Identity management of cloud systems	Uses encryption data for enabling multi-party communications that include both trusted and untrusted entities. This implementation also looks at helping service providers and hence cannot be deployed for identifying suitable service providers based on their performance.
-------	--------------------------------------	--

From the above discussion, it can be seen that the cloud trust management systems are suffering from several shortcomings including the fact that many of the proposed models lack prototype implementation for proof of concept. Many of the proposed models are incomplete and do not meet all the requirements of a cloud systems. In addition, except for the fuzzy theory based model proposed in [50], all the others lack a firm theoretical foundation to base the implementation on solid footings. Thus, in this research fuzzy theory based trust computing mechanism would be used as the basis for developing an improved trust computing mechanism.

2.5 Summary

This chapter mainly concentrated on a critical analysis of published literature in the broad areas of cloud computing, quality of service and trust computing. The main purpose of this literature review is to understand the work already carried out in the area and to find the research gap for formulating the guidelines for continuation of the work. The chapter summarizes the entire literature review under various properly selected sections and subsections for clarity.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

The primary objective of this research project is to come up with an adaptive trust based quality of service monitoring mechanism for cloud computing. The overall design of the mechanism is expected to have three submodules meeting the objectives presented in Chapter One. The mechanism developed as part of this project must be evaluated with a view of verifying and validating its operation to ensure that it achieves the objectives set out in Chapter One. It is important that the research project has been carried out according to well defined plan, so that the methods adopted at various stages of the project are scientifically valid and can be verified to produce similar results, if repeated under same conditions. This chapter presents the research methodology adopted in this work in order to achieve the said objectives in a valid and verifiable way. The Design Research Methodology (DRM) [150] proposed by Blessing and Chakrabarti provides the basis for the methodology presented in this chapter. The proposed research methodology incorporates the modified or adapted stages of the DRM in order to make them better suited for this research project. The proposed research methodology consists of four phases as Analysis, Design, Testing, and Evaluation as shown in Figure 3.1. The DRM, in its basic form also consists of four main stages; research clarification, descriptive study I, prescriptive study and descriptive study II. Figure 3.2 shows the stages of the Design Research Methodology along with the relationship between the stages, the means adopted at each stage and the main outcomes.

Section 3.1 discusses the overall research approach adopted in this research project for achieving the research objectives formulated in Chapter One. Section 3.1

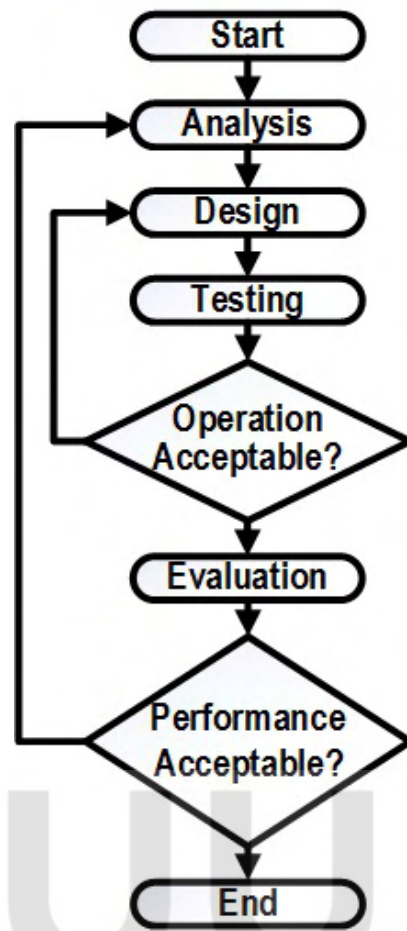


Figure 3.1. Research Methodology

further explains in depth how the DRM has been modified to suit the requirements of this project along with specific approaches employed in each stage along with the corresponding deliverables. Section 3.2 presents the first stage of the DRM methodology namely, Research Clarification (RC), along with the modifications that have been incorporated for the purpose of meeting the requirements of this research project. The discussion centers on the aims of this stage, methods used in this stage and the deliverables. Section 3.3 describes the adapted version of Descriptive Study I (DS-I), the second stage of DRM with special emphasis on understanding the current status of technology, design and propose a conceptual model based on the gap identified. The methods adopted in designing the trust based quality of service monitoring mechanism are presented in Section 3.4 under the Prescriptive Study (PS) phase of the DRM. Section 3.5 highlights the performance evaluation methods employed in this project

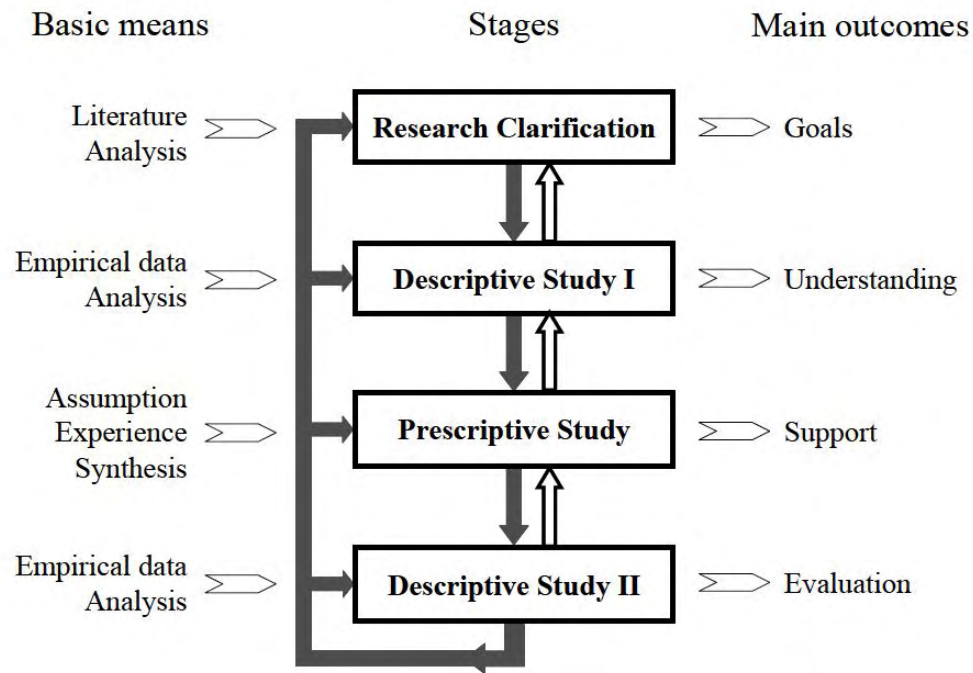


Figure 3.2. Stages of Design Research Methodology [150]

along the discussion of metrics used. Finally, Section 3.6 concludes the chapter providing a comprehensive summary of the chapter.

3.2 Research Approach

The main objective of this research project is to develop a quality of service monitoring mechanism based on trust computing principles that can adapt itself to the changes in the behavior of the service providers. By adapting to the changes in the behavior of service providers, the monitoring mechanism will furnish the most recent and accurate information on the ability of them to meet the demands of the customers. The research approach adopted in this project must be scientific and comprehensive enough to guide the entire process from the beginning until the end, so that the experiments conducted and results produced are trustworthy and repeatable in comparable environments.

The methodology adopted in this research consists of four main phases. They are namely analysis, design, testing and verification & validation. In order to make the

research approach scientifically strong and valid, it was decided to adopt features from the well established and scientifically proven DRM into the proposed research methodology [151, 152]. The DRM can be considered as an approach, guideline or a framework of supporting methods that enables the design research to be more rigorous, effective and efficient making the outcomes valuable both academically and practically [150]. The main objectives of DRM are as follows:

- to provide a framework for researchers to conduct design research efficiently;
- to help researchers to identify research areas that are both academically and practically challenging and useful;
- to enable researchers to select and combine more than one research methods effectively; and
- to provide guidelines for rigorous and systematic research planning.

Figure 3.3 shows the different phases of the research methodology adopted in this work in the light of DRM stages along with the methods used to achieve the objectives of each phase/stage and the expected deliverables. The main phases of the research methodology are analysis, design, testing, and verification and validation. The corresponding DRM stages are research clarification, descriptive study I prescriptive study and descriptive study II. The process flows between the stages are shown in narrower arrows while the thicker arrows associate the methods used with the different stages of the DRM to the deliverables at each stage. The following sections provide brief explanations for each phase/stage in conjunction with the explanation of methods and deliverables.

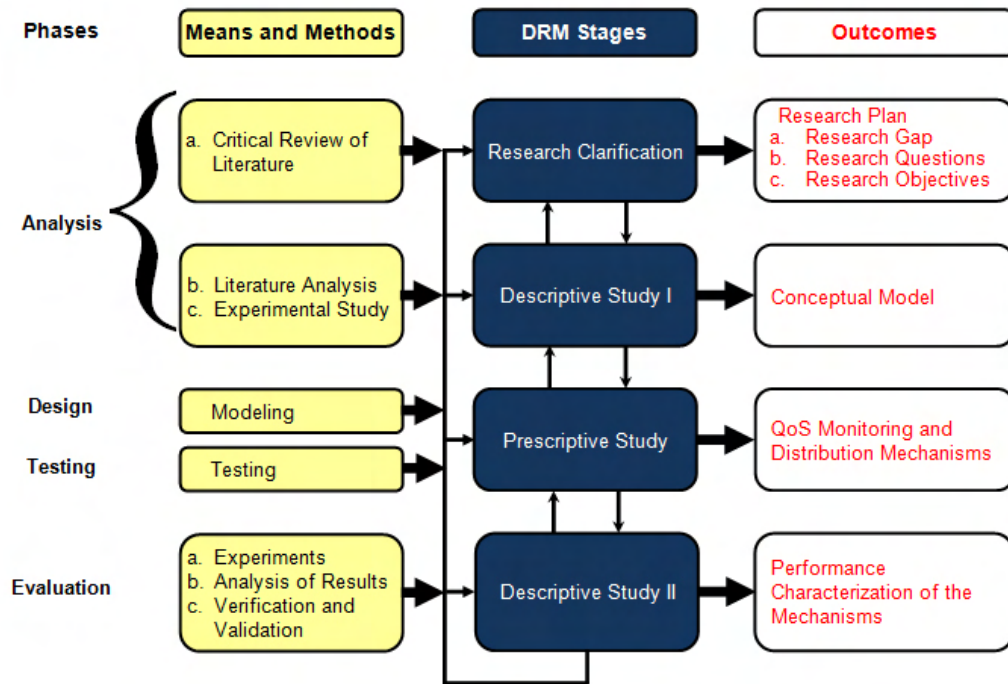


Figure 3.3. Research Approach

3.2.1 Analysis

The analysis phase of the research methodology is made up of the combination of both clarification and descriptive study I of DRM as shown in Figure 3.3. This phase mainly concerns with obtaining a clear understanding of the research project including the background, research problem, gap, questions and objectives culminating with the development of a conceptual model. The methods adopted at this phase include critical review of existing literature and experimental analysis of existing mechanisms, in any. Further details of this phase are discussed in the first two stages of DRM.

3.2.1.1 Research Clarification

It is necessary to obtain a clear understanding of the research to be conducted at the beginning itself. By obtaining a clear and precise understanding of the current status of research area selected, it would be possible to define the goals of the research project with a challenging but realistic project plan. The research clarification stage

can be further divided into six steps as shown in Figure 3.4. These steps are inherently iterative helping to improve the results towards the objective gradually.

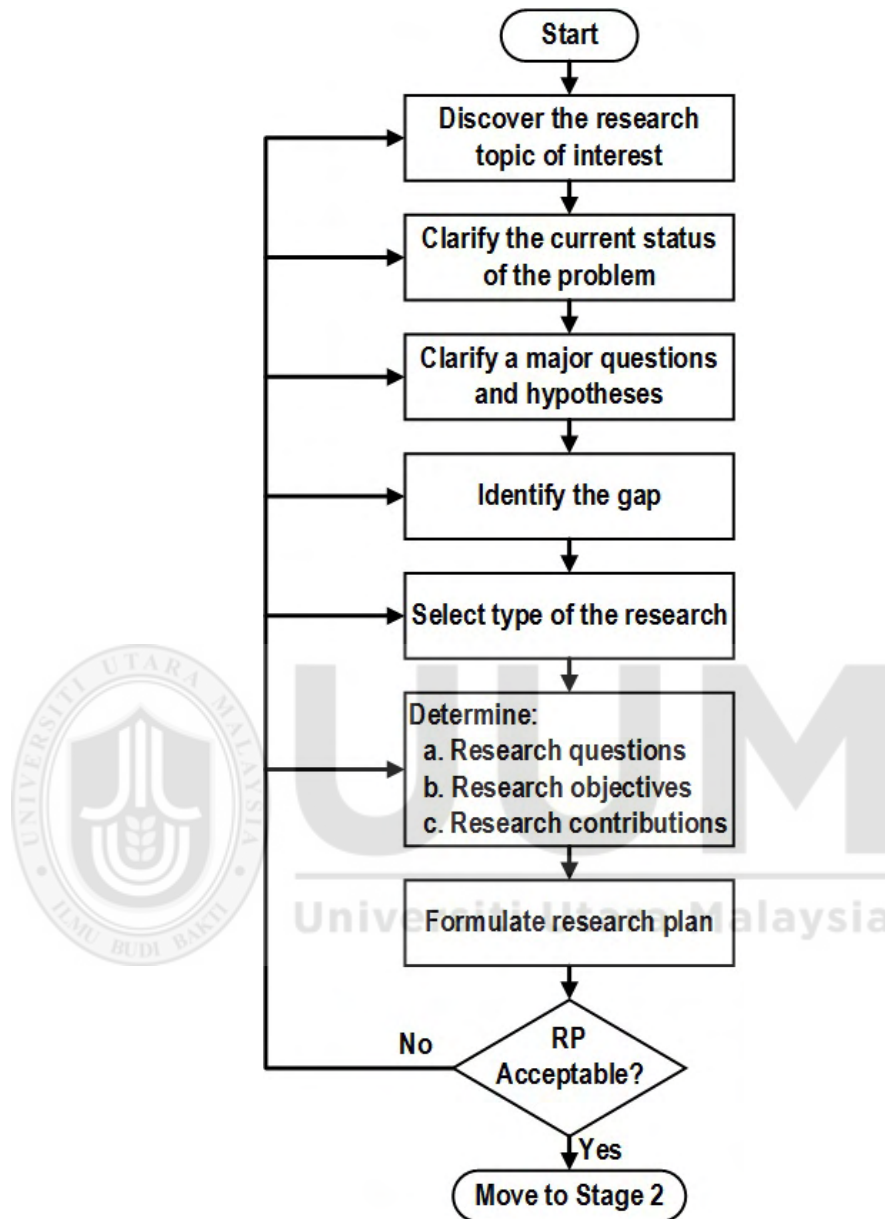


Figure 3.4. Main Steps Involved in Research Clarification Stage

The method adopted at the research clarification stage is the analysis of the existing published literature in the chosen area. At the end of this phase, the researcher would have acquired a broad and in depth knowledge in the chosen field especially the strengths and weaknesses of the current approaches and mechanisms. The weaknesses of the current approaches will be exploited to identify the research gap and then formulating research questions and objectives.

The main deliverable at the end of research clarification stage is Chapter One of the thesis. In addition to the main deliverable, a research (survey) paper could also be produced as subsidiary outcome. The research plan formulated at the conclusion of this stage would include the following:

- research focus and motivation,
- research problem and questions,
- research objectives,
- areas to be studied in depth,
- research approach including type, scope, important milestones, methods to be adopted and contributions.

3.2.1.2 Descriptive Study-I

The Descriptive Study I (DS-I) makes the second part of the analysis phase of the research methodology. During the DS-I stage, the research area identified in the RC stage would be further investigated with the view of obtaining an in depth understanding of the current status of the specific area selected. The method adopted at this stage would include critical review of the literature along with experimental analysis of the existing approaches, where possible. During the course of this research, an in depth study on the existing solutions was carried out [153, 154, 155, 156]. DS-I is made up of five steps as shown in Figure 3.5, which must be applied recursively improving the understanding of the process further and deeper. The conceptual model to be developed as part of the DS-I stage will also be refined and improved at the application of each step as the deeper understanding of the shortcomings of the current techniques or mechanisms would work as a catalyst for developing better solutions.

The expected deliverables at the end of DS-I include:

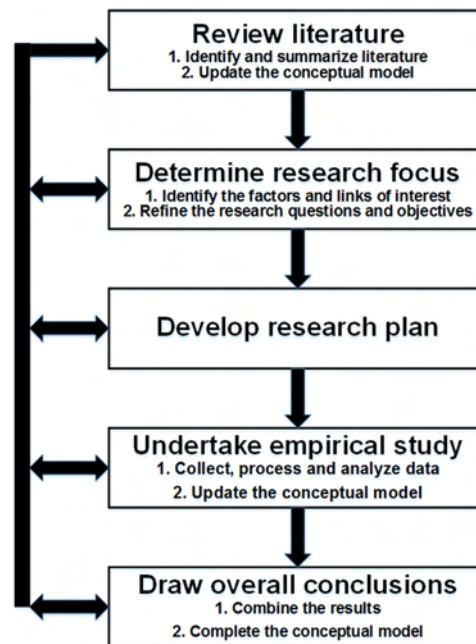


Figure 3.5. Main Steps in Descriptive Study - I

- critical review of quality of service monitoring in cloud computing as presented in Chapter Two and
- conceptual model for the proposed service quality monitoring mechanism.

Conceptual Model

In order to make cloud computing more useful and acceptable by customers, it is necessary to increase the users' confidence in the technology and the service providers. Prior to adopting this technology as their primary mode to computing services, clients are required to select the right service provider, who will meet their requirements in terms of cost as well as performance [157]. Hence it is necessary to have a mechanism that can effectively monitor the service quality of the providers and quantify it in such a manner that users can select the most suitable cloud provider. In order to meet this requirement, the overall objective of this research has been formulated as to develop an effective service quality monitoring mechanism for cloud computing. Three sub-objectives have then been identified with the aim of simplifying the tasks. The sub-

objectives being:

- i. to develop a technique for modeling and quantifying service quality to a single score,
- ii. to formulate a mechanism to dynamically adjust the service quality score, and
- iii. to devise a method for distributing the quantified service quality score between cooperating monitors.

For the purpose of attaining the above said objectives, a conceptual model of the research has been formulated. Figure 3.6 shows the conceptual model developed at the end of Descriptive Study-I.

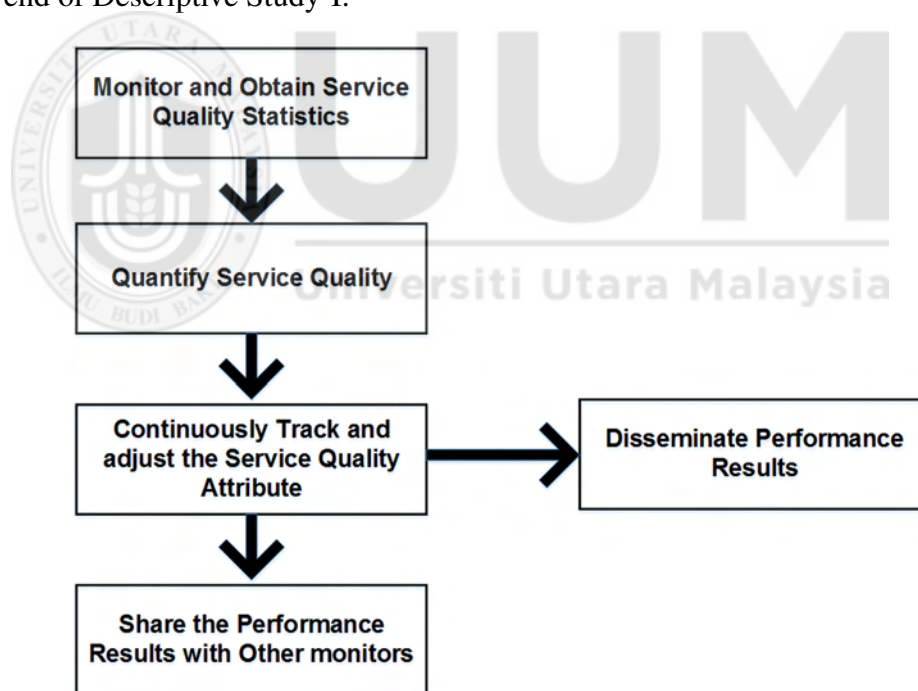


Figure 3.6. Conceptual Model

3.3 Design

Phase 2 or the design phase of the research methodology starts from the conceptual idea formulated in Phase 1 for solving a research question and ends with the development of technique or mechanism. Until the conceptual idea formed in Phase 1

matures to a complete mechanism that can be functional tested, the processes in this phase must be repeated.

The design phase of the research methodology makes the upper part of the Prescriptive Stage of the DRM as shown in Figure 3.3. For the purpose of this research, an iterative model, design, test and integrate approach has been selected. Figure 3.7 shows the steps involved in the mechanism development process.

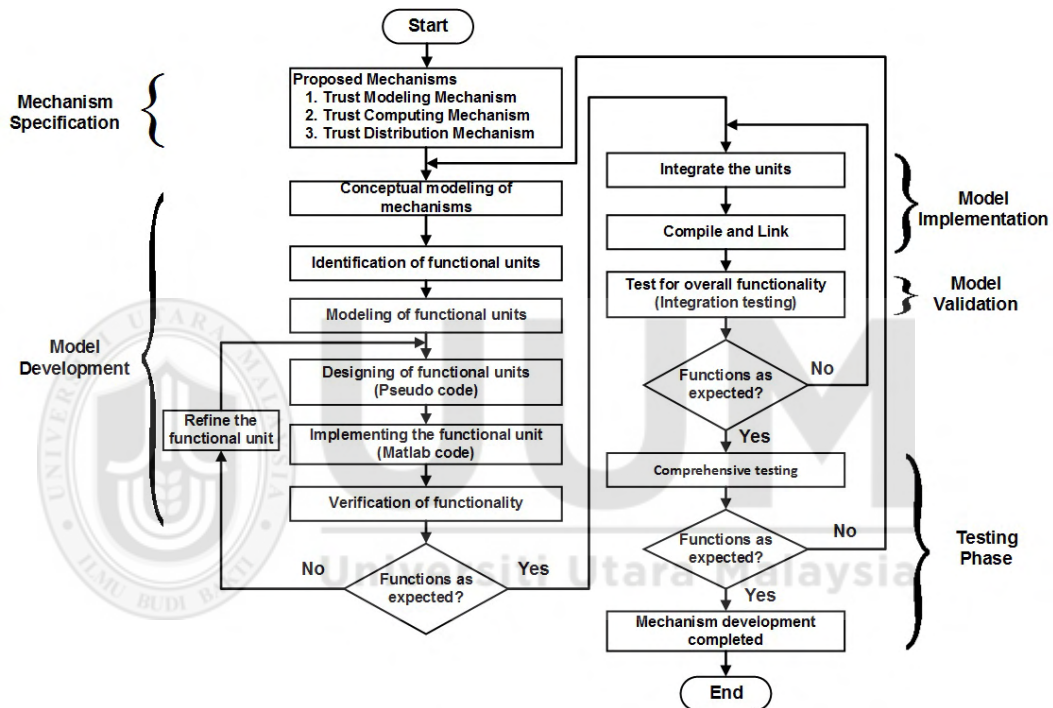


Figure 3.7. Mechanism Development Process

The main steps of the design phase (prescriptive study stage) include the specification of the proposed mechanisms, model development, model implementation and model validation as shown in Figure 3.7. The model development process is generally a complex operation with many sub-processes (sub-steps) that must be traversed many times iteratively. Many assumptions and simplifications may be made at this stage depending on the complexity of the model in order to arrive at realistic design. Model implementation is basically the integration of the sub-units as a single, large and self-contained unit that can carry out a defined task independently and testing for the overall

functionality. The model implementation involves coding and building (compiling and linking) on the chosen simulation environment. The model validation is the integration testing process that must be carried out to ensure the proposed mechanism fulfills the intended objectives. Detailed discussion these sub-processes are given in the following subsections.

Deliverables at the end of the design phase are:

- modeling of the proposed mechanisms,
- design, implementation and validation of the model,
- Chapters Four, Five and Six and
- several research papers.

3.3.1 Model Development

From Figure 3.7, it can be seen that the mechanism development process is essentially an iterative process going through several refinements until a satisfactory mechanism has been arrived at. At the beginning of this process, the conceptual idea must be broken down into self contained functional units. These individual units must be capable of accepting inputs, process them and produce outputs that can be further processed by other downstream functional units. Each individual unit must be modeled and designed using the most appropriate technique including mathematical, statistical, iterative or any other. Once a functional unit has been fully modeled, it must be tested for its operation. Operational testing of the units can be carried out either through a manual or automated process. Manual testing of functional units is generally carried out through a paper-pencil method, where the operation of the unit is computed using user controlled inputs and results are generated. Manual testing of each unit for the entire input range would be very tedious and very error prone. There is also a

chance of missing abnormal behavior of the units for certain specific input values as the testing would only consider a sample of all the possible inputs. Automatic testing of the functional units can be carried out by designing an appropriate computer algorithm and implementing it using computer software. The automatic testing of the units would be more comprehensive, accurate and efficient compared to the manual testing as it is possible to consider the entire range of inputs in a shorter time. The output can also be better visualized with the aid of graphical utilities such as Gnu Plots with two or three dimensional plots. Presenting outputs in graphical formats would help understand the operation easily and detect abnormal operations faster. The downside of the computer based testing the difficulty of implementing complex mathematical operations represented by differential equations, Fourier transforms etc. This shortcoming can be overcome by using an advanced analysis software like Matlab using built-in libraries [158]. The built-in libraries are more accurate and easier to implement than functions implemented using general purpose computer languages.

The modeling, designing and testing of every functional unit must be carried iteratively until the desired functionality is achieved. Whenever a functional unit does not perform as expected throughout its input range, it must be troubleshoot and fixed. This must be continued until all the units are working the way expected.

3.3.2 Model Implementation

Once all the units are individually implemented and tested, they all must be integrated into a single unit forming the fully functional mechanism. If the individual units have been implemented in a modular fashion, it is possible to integrate them into a single unit with little effort. Thus, it is very important from the beginning to follow good programming practice as it will save a lot of time and effort towards the critical stages of the research. Modules can be implemented as either functions or objects

(classes) depending on the type of programming language selected. Advantages of modular programming include easy to write, understand, debug, maintain and integrate them together to form a large complex program [159]. Also modular programming facilitates code reuse, as a generic function can be called more than once by passing attributes at run time. Once the program has been developed in a modular fashion, the main program will look neat and organized containing only few lines calling these modules.

In this research, modules were implemented as functions in Matlab. In Matlab individual self contained units can be coded as modules and stored in separate M-files [160]. Since Matlab is an interpreted language, it makes coding and debugging faster compared a compiled language. The interpreted computer languages allow testing of the program codes immediately without going through the steps of compiling and linking for creating the executable code. The other advantages of Matlab include the way it handles input and output. Input can be wither dynamically generated using a function or stored in a text file and called at run time. Similarly the output can be directed to a file stored in the permanent storage and/or displayed on screen as raw data or graphically in the form of 2-D or 3-D plots using the integrated visualization utilities. These graphs can be customized either interactively or programmatically. The complete mechanisms were implemented in separate M-files, which in turn called the individual functional units stored in separate M-files.

3.3.3 Model Validation

Model verification determines whether the model has been transformed from one form to another with sufficient accuracy [161]. In other words, the model verification process evaluates and verifies the accuracy of the porting of an application from a pseudocode or flowchart to an executable computer program developed in a high level

language. Model validation is the process of determining the degree of consistency of a computer model with respect to the real world phenomenon or application [162]. Model validation is a compulsory procedure required to be carried out in any research project before conducting any substantial experiment using computer models in order to ensure that the methods and mechanisms meet the intended requirements and the results obtained are valid.

A mechanism must be considered as a single unit and tested again for the entire input range. Though, the individual functional units have been tested and proven to be working as expected, there can still be problems when they are integrated together [163, 164]. The main issues that might arise at this stage include functional mismatches between subunits (outputs to inputs), timing errors etc. These errors must be caught and rectified through implementing check points at important places. Once the check points established, testing the overall implementation must be carried out using random inputs covering the entire range. If the final outputs along with the readings at all check points are acceptable, then the mechanism can be concluded to be working as planned. Otherwise, the entire process must be restarted from the conceptual design itself.

In this project the modal validation (integration testing) is carried out at the end of the integration of all the functional units into a single module. The initial test data were created using simple mathematical functions to check the interoperability of the functional units. The interfaces between the units were used as breakpoints to check the integration between the units. The values at the breakpoints were checked and verified using paper-pencil (manual) checking method for some specific input values. This test has been repeated several times using different test cases in order to make sure that all the submodules are working as intended and they interact properly through the

right interfaces.

3.4 Testing

Testing, the Phase 3 of the research methodology focuses on the verification and validation of the mechanisms developed in Phase 2. The testing phase of the research methodology makes the second half of the Prescriptive Study (PS) of the DRM as shown in Figure 3.3. As shown in Figure 3.7, the comprehensive testing has been carried out at the end of the model validation process. Since comprehensive testing is strongly connected with the model development, implementation and validation process, it has been drawn in the same diagram in Figure 3.7.

In order to test the operation of the mechanisms, they must be ported to an environment that is close to the real world settings. Hence the mechanisms were ported to CloudSim a popular simulation environment specifically designed for testing cloud computing operations [165]. CloudSim has been selected for this purpose as it provides a generalized and extensible simulation toolkit and application that enables seamless modeling, simulation, and experimentation of emerging cloud computing system, infrastructures and application environments for single and federated clouds [166, 165]. Also CloudSim provides the best and the most sophisticated simulation environment compared to many other cloud computing simulators in the market such as CDOSim, TeachCloud, iCanCloud, SPECI, GroudSim, DCSim etc [167, 168, 169, 170, 171, 172, 173, 174]. Also the developers of CloudSim have developed two more simulators based on CloudSim known as Network CloudSim and CloudAnalyst enhancing the scope of cloud simulation [175, 176]. Thus CloudSim along with CloudAnalyst and Network CloudSim provides a complete simulation environment, where it is possible to focus on specific system design issues without getting concerned about the low level details [175, 177]. Hence in this research CloudSim has been

selected as the simulator platform for analyzing of the mechanisms developed. The details of CloudSim simulator will be discussed in Section 3.5.2.1.

The proposed mechanism were implemented in Java programming language as they will be tested using the CloudSim simulation framework that has been implemented in Java [165, 178]. In order to make sure that programs written are syntactically correct and free of errors, the code must first be verified using an automated code verification tool [179]. QJ-Pro has been used as the code analysis tool to verify that the computer programs have been correctly coded and free of bugs in this research. Since the CloudSim is implemented using individual files and libraries and lack an Integrated Development Environment (IDE), the Eclipse IDE for Java Developers (Version: Mars Release (4.5.0)) was used in this work. The IDE provides a complete development environment comprising the code editor, debugger, launcher, parser, compiler and linker in one single tool along with a user friendly interface. The possibility of integrating QJ-Pro into the Eclipse IDE as a plugin is an added advantage. Figure 3.8 shows the Eclipse Integrated Development Environment for Java installed with CloudSim package.

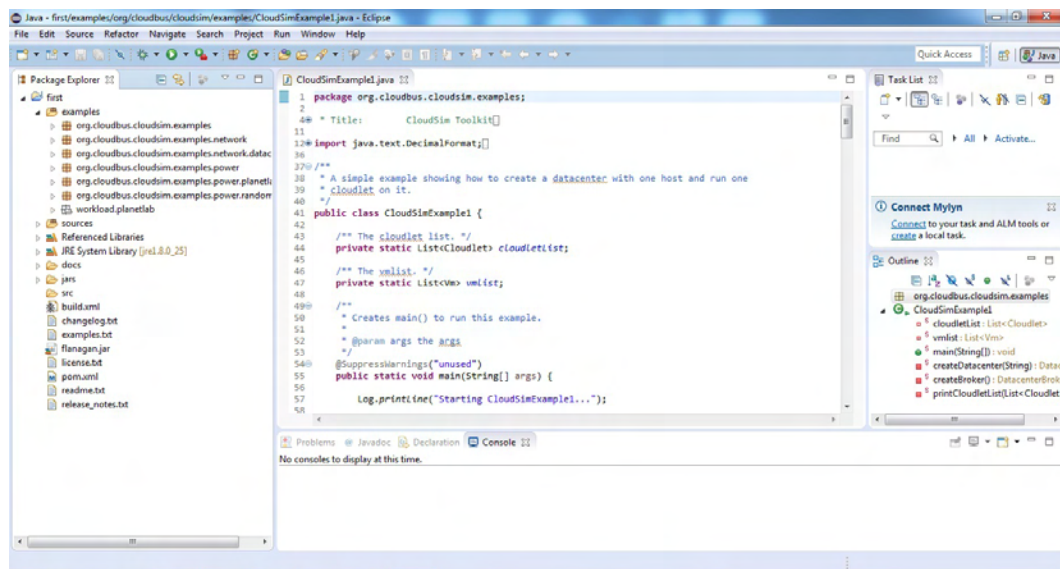


Figure 3.8. Eclipse Integrated Development Environment for Java

Code analysis with QJ-Pro provides many advantages over manual inspection of code for errors. The QJ-Pro drastically reduces the code review time by automating the code inspection and coding standard enforcement process. QJ-Pro provides the developers with the ability to automatically evaluate the software on the basis of the advanced software quality concepts such as reliability, maintainability and efficiency. Figure 3.9 shows the code analysis window of the QJ-Pro loaded with a sample file.

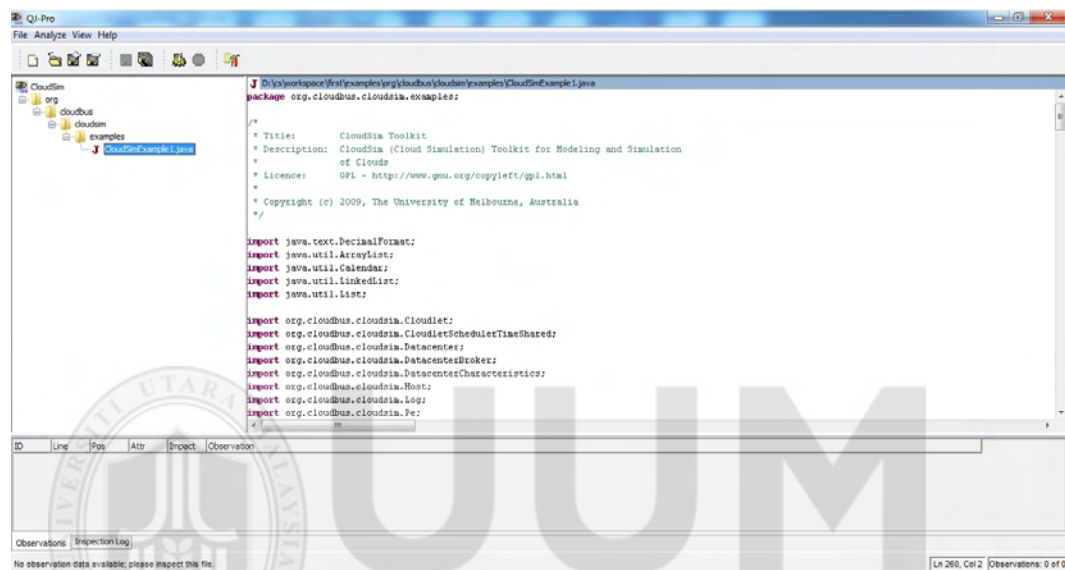


Figure 3.9. QJ-Pro Code Analysis Window

Verification and validation techniques can be divided into four groups as formal, informal, static and dynamic [161]. The dynamic analysis is carried out to test the physical behavior of the model to a set of time varying inputs. Hence the software application must first be executed and the input fed to the model. The dynamic test is capable of detecting potential software errors not found by static code analysis or any other traditional testing methods [180]. Hence in this project, dynamic code analysis was employed as the primary validation technique. Dynamic code analysis requires well formed and defined input data for the analysis to work correctly. The dynamic analysis include several functionalities including the analysis of execution trace, call stack, allocation and deallocation of objects, cycle counts etc [180].

The dynamic analysis process generally goes through three specific steps for validating the software code, namely instrumentation, execution and analysis of output [181]. Instrumentation of application refers to adding additional code into the program for collecting information related to the model behavior during the execution. Code execution is the actual running of the program with the dynamic input supplied. And finally the output data is collected and analyzed by comparing it to the output of other models. The last step is commonly known as benchmarking.

The main objective of this research is the development of mechanism for monitoring, quantifying and distributing service quality of cloud service providers. The model proposed in this chapter was used as the basis for the development of the mechanisms. The mechanisms developed in this project were verified and validated using the simulation implemented in CloudSim.

3.5 Evaluation

Evaluation is the last phase of the research methodology adopted in this research. The main focus of this phase is the comprehensive evaluation of the mechanisms developed. Performance evaluation is one of the most important steps in any research project. The performance evaluation of QoS monitoring, quantifying and distribution can be carried out using one of the three possible techniques as analytical modeling, simulation or testbedding [182, 183, 184]. Evaluation phase encompasses the entire Descriptive Study-II (DS-II) stage of the DRM as shown in Figure 3.3.

3.5.1 Selecting the Evaluation Approach

The selection of the appropriate evaluation approach is a crucial step in any research project [185]. As part of this project, the different evaluation approaches have been compared for their strengths and weaknesses along with their suitability for the

application in this work. Table 3.1 lists the strengths and weaknesses of each approach with respect to the application in computer network research.

Table 3.1

Comparison of Different Evaluation Approaches Adapted from [185]

Criteria	Analytical modeling	Simulation	Test bedding
Time required	Low	Medium	High
Accuracy	Low	Moderate	Highest
Tools	None	Computer Software	Real Equipments
Trade off evaluation	Easy	Moderate	Difficult
Cost	Lowest	Moderate	Highest

3.5.1.1 Analytical Modeling

Analytical modeling or commonly known as mathematical modeling is a set of equations formulated using mathematical concepts to describe behavior of a physical system [186, 187]. The mathematical model thus developed can be used to analyze the performance of the system for various input conditions. The evaluation can be done either manually or automatically using a computer. Manual analysis may be cumbersome and take long time for some models depending on complexity of them. Using computers has become the common practice for solving analytical models due to the availability of advanced software packages equipped with libraries that can solve complex mathematical equations and function with minimal effort and custom configurations [188]. The results of an analytical model can be presented in many forms including tables, graphs etc [189]. This makes visualization and comprehension of the results easier. It is possible to adjust the condition of the system varying the input or parameters with relative ease. This method is very suitable for studying the behavior of dangerous and unsafe systems. This provides an opportunity to obtain an initial view or understanding of a system before embarking on building a prototype for further study or the complete system.

The main shortcoming of this method is its difficulty to build a complete system with mathematical models alone when the system becomes more complex with many interrelated units. Hence, when the complexity of a system increases it becomes necessary to make simplifications and assumptions to focus only on certain aspects of the system and to make the rest static. The advantages and disadvantages of analytical modeling include low cost, easier trade-off evaluation and low cost [190]. However analytical modeling accuracy is lower compared to other methods especially when the system becomes more complex.

3.5.1.2 Testbedding

Testbedding is the method of implementing a prototype of an actual network at lower complexity and scale with the objective of running tests and collecting results [191]. Testbeds can provide very accurate results as they mimic the real world scenarios using real equipment, but they are more difficult and expensive to construct [192, 191]. Also it is very unrealistic to build a cloud computing testbed with hundreds or thousands of computing nodes and control them to obtain real world data. If the limited sized testbed is built with few nodes, it would limit the scope of the experiments and the results obtained would not be realistic. The repeatability of the experiments may also be an issue with testbeds [165]. Commercial organizations such as Yahoo Inc., Intel Corporation and Hewlett Packard Limited have launched a cloud computing testbed spread across the United States, Singapore and Germany called Open Cirrus for conducting research in a real environment [165]. But, access to this infrastructure is limited only to their member organizations. Building a cloud infrastructure of this nature is beyond the scope of this research project.

3.5.1.3 Simulation

Simulation has been a widely used technique for analyzing the behavior of dynamic systems lately [165]. Simulation is carrying out experiments using computer based system models that have been developed using complex equations and algorithms. Simulation provides a flexible environment for carrying out in-depth study on protocols and mechanisms. The repeatability of experiments is very high with simulation. Also, with simulation, it is possible to study the effect of specific parameters on the output by changing only those parameters while keeping all the others constant. This enables the analysis of the performance of protocols and mechanisms in a scalable, controllable and repeatable environments [193]. Due to this reason, simulation has been widely used in the performance evaluation and validation of results in computer networking research [183, 165, 194]. Hence simulation has been selected as the main evaluation technique in this research. The main advantages of using cloud simulators for evaluation of results in research include their capability for efficient modeling and simulation of [165, 195, 196, 197]:

- large scale Cloud computing data centers and federated clouds,
- virtualized server hosts, with customizable policies for provisioning host resources to virtual machines,
- energy-aware computational resources,
- data center network topologies and message-passing applications,
- dynamic insertion of simulation elements, stop and resume of simulation, and
- user-defined policies for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines.

3.5.1.4 Evaluation Environment

As simulation has been the primary evaluation technique in this research, it is imperative to select the right simulation tool. There are many cloud computing simulators in the market with different features and capabilities. Some of the popular cloud computing simulators include CloudSim suite, CDOSim, TeachCloud, iCanCloud, SPECI, GroudSim and DCSim [165, 169, 170, 172, 173, 174, 198]. Each of these simulators has its own strengths and weaknesses. A critical study on literature published about these tools was carried out in order to identify the right tool for this research.

CDOSim [169] is a simulation tool for simulating the cost and performance of Cloud Deployment Options (CDO) especially cloud service providers. CDOSim is built extending the CloudSim simulator and integrated into another cloud migration framework known as CloudMIG. CDOSim performs well in predicting the cost and performance characteristics of CDOs when checked against both private as well as public cloud computing environments. The main shortcoming of this simulator is its inability to work as a general purpose cloud simulator accommodating other parameters such as response time, job completion time etc.

TeachCloud [170] is a simulation tool developed specifically for teaching cloud computing in a classroom environment. This tool provides an excellent environment for creating and experimenting with various cloud computing components such as processing elements, data centers, networking, SLA constraints, applications, virtualization, management and automation, and business process management. But the usefulness of this tool in research is limited as it lacks provision for modification of the behavior of the basic modules.

iCanCloud [198] simulation platform has been designed with the objective of predicting the trade-off between the cost and the performance of a specific cloud based application in a given environment. It has been developed by extending the SIMCAN simulation framework [199]. It is possible to develop applications in iCanCloud using traces of real applications, state graphs and directly in the simulation platform. However, it does not support importing existing applications directly and they must be modeled manually. This is a simulation tool with very narrow scope concentrating only on the application cost-performance trade off. Hence it is not suitable for this research.

Simulation Program for Elastic Cloud Infrastructures or SPECI [172] in short is capable of analyzing the different scalability and performance aspects of data centers. When data centers grow with time, this growth happens in a non-linear fashion. Hence it is necessary to analyze the behavior of such data centers. SPECI has been specifically designed to simulate and predict the behavior of these growing data centers, so that the service providers can plan their data center operations and enhancements in advance. This simulation tool also developed targeting a specific user group such as cloud service providers in order to help them plan and design their data centers properly. This tool lacks many features making it unsuitable to be used in general purpose cloud computing research. Hence this tool has not been selected in this research.

GroudSim [173] simulation tool is capable of simulating both cloud and grid environments. GroudSim provides a comprehensive set of features for extensive simulation of scenarios from simple job execution on leased computing elements to the calculation of cost and background load on those elements [200]. One of the main features of the GroudSim is its ability to incorporate failures within the simulation.

Simulation environment can be parameterized and extended using probability distribution packages for the inclusion of failures within cloud elements. GroudSim has been designed specifically for modeling and testing scientific workloads. This may be due to the influence of the grid computing elements implemented within the simulator. This becomes the main shortcoming of the GroufSim package for general purpose simulation implementations, where various different workloads with competing requirements need to be tested.

A Data Center Simulation Tool for Evaluating Dynamic Virtualized Resource Management, in short DCSim [174] is an extensible framework for simulating a multi-tenant virtualized data centers. The main focus of this simulator is the data center management techniques such as reallocation of virtual machines in order to optimize the resource utilization and reduce overall cost. Though DCSim can simulate the management of virtual machines within a data center effectively, it cannot simulate federated cloud data centers where multiple geographically distributed data centers operating collaboratively by coordinating resource allocations between them. Hence CDSim is also not a suitable tool for this research, as it is necessary to evaluate the distribution of trust scores among cooperating QoS monitoring and quantifying agents.

CloudSim suite [165, 175, 176, 201] is a collection of cloud computing simulation tools that have been developed with the objective of evaluating different strategies, mechanism, algorithms and protocols.

Table 3.2 shows a brief comparison of different open source cloud computing simulators available in the market.

Table 3.2

Comparison of Different Cloud Simulators [202, 203, 167, 169, 170]

Simulator	Programming environment	Federation support	Simulation time	Comments
CDOSim	Java	No	Minutes	Only cost and performance characteristics can be simulated.
TeachCloud	Java	No	Minutes	Designed for teaching purposes only.
iCanCloud	Java	No	Minutes	Only Amazon instance types are supported
SPECI	Java	No	Seconds	Only single isolated data centers can be simulated
GroudSim	Java	No	Seconds	Specifically designed simulating scientific applications in cloud and grid environments
DCSim	Java	No	Minutes	Only a data center simulation tool
CloudSim	Java	Yes	Seconds	Complete suite with a many facilities for extensive simulation.

From Table 3.2, it can be seen that CloudSim would be the most suitable simulation tool for studying the different aspects of large distributed cloud systems. Also CloudSim has been considered as the most sophisticated cloud simulation tool available in the market today [167]. CloudSim has also been used as the underlying platform for many other cloud simulation tools such as SmartSim, CDOSim, TeachCloud and EMUSIM [203, 202]. Hence CloudSim has been selected as the simulation tool in this research.

3.5.2 CloudSim Simulation Suite

CloudSim simulation suite has been made up of several simulation tools that enable the simulation of various aspects of large distributed cloud systems [177]. The following subsections briefly look at each one of them individually.

3.5.2.1 CloudSim

CloudSim has been developed by one of the leading research groups in cloud computing based at the University of Melbourne, Australia [197, 165]. It supports the modeling and simulation of large cloud computing infrastructure with many data centers on a single computer and Java virtual machine. It has the capability for specifying many different cloud computing resources including users, data centers, service brokers, scheduling and allocations policies. All these resources have been implemented as Java classes that can be easily extended to suit different requirements. These components can be arranged together making different architectures, where users can evaluate new policies, algorithms, mechanisms and mapping before deploying them in real world systems. It has the capability of putting together most of the real world cloud scenarios by simply extending or replacing the classes and coding the desired scenario. The main shortcoming of this simulator is its lack of a Graphical User interface (GUI).

CloudSim Architecture

CloudSim has been designed in a modular fashion with a layered architecture [197, 165]. Figure 3.10 shows the CloudSim layered architecture along with different modules incorporated with the layers. The bottom two layers namely the SimJava and GridSim layers provide the necessary core functionalities required by the upper levels for implementing operations like queuing, processing of events, creation and removal of system components, communication between components, managing the simulation clock, creating and managing networks along with their traffic profiles.

The CloudSim layer programmatically extends the functionalities of GridSim for enabling the modeling and simulation diverse cloud computing environments. This layer is responsible for the creation and management of cloud computing resources

such as datacenters, hosts, virtual machines, applications etc. This layer has the ability to instantiate thousands of components concurrently creating large scale cloud systems.

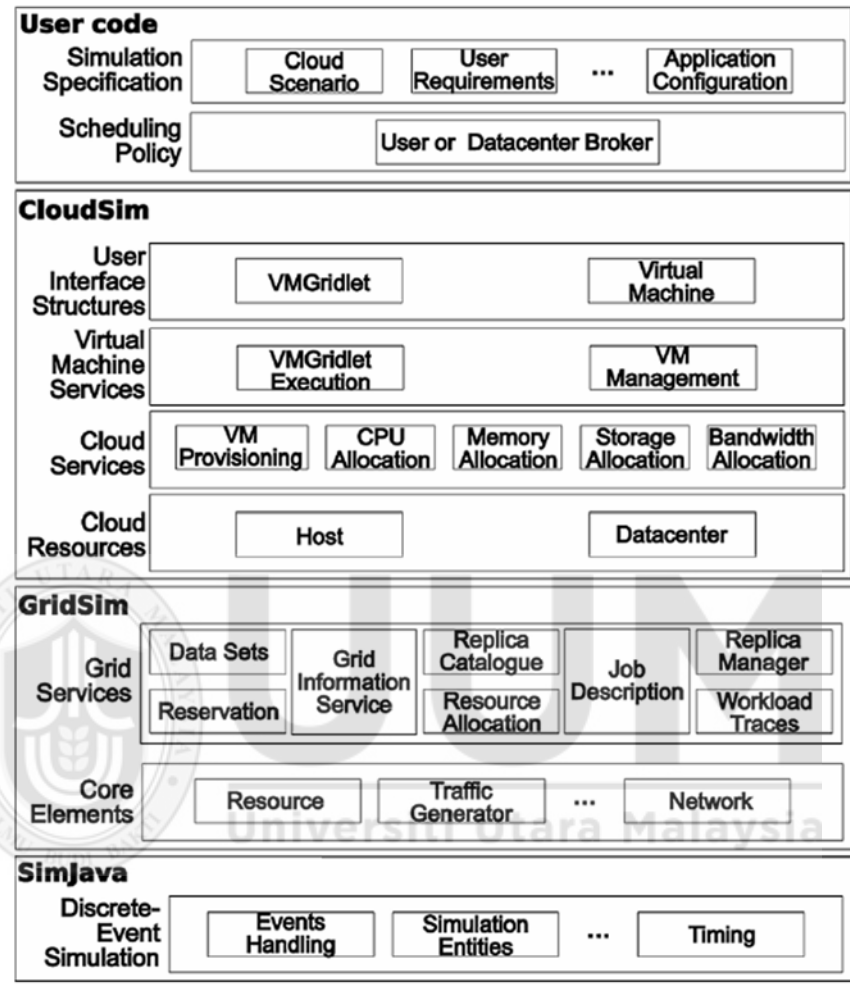


Figure 3.10. CloudSim Layered Architecture [165, 166, 197]

The top-most layer holds the user code for creating and customizing the simulation environment required by him. This layer allows users to manage the configuration related functionalities for all the components including datacenters, hosts, virtual machines, applications, user policies etc. The different components within CloudSim communicate with each other by the way of message passing between themselves. Figure 3.11 shows the CloudSim class diagram containing the fundamental classes that make the building blocks of the simulator.

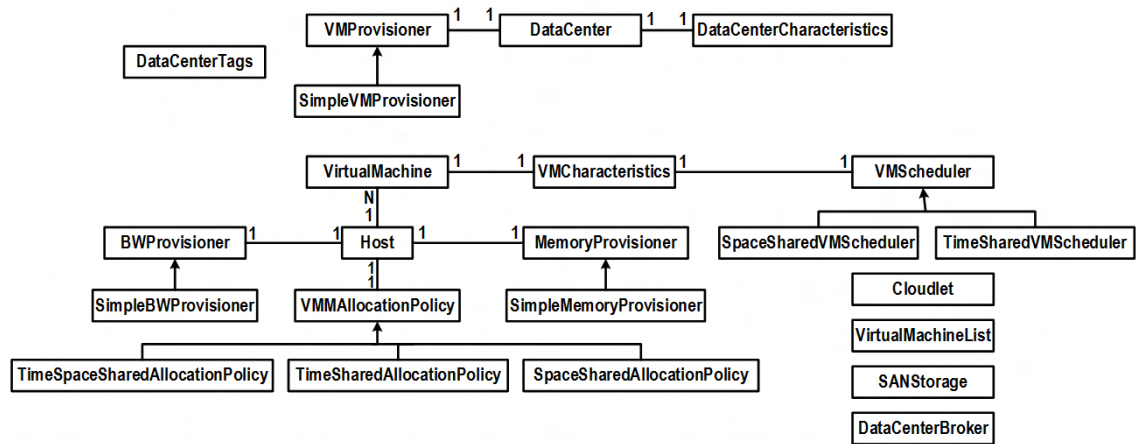


Figure 3.11. CloudSim Class Diagram [165, 166, 197]

Modeling Cloud Systems with CloudSim

Large federated cloud systems can be easily modeled with CloudSim by extending the entities within the simulator programmatically. The infrastructure level service related to the clouds can be modeled and simulated by extending the datacenter entity. The datacenter entity manages and controls the host entities which are assigned with virtual machines according to a predetermined policy. The assignment of virtual machines to physical hosts needs to consider several factors such as availability of physical resources, the demands of the virtual machines to be created, type of allocation (space shared or time shared) etc. All these need to be carried out through extending the VMMAAllocationPolicy class. Similarly other core classes can be extended to create the entities such as cloud market managing the cost-benefits of using different cloud systems, network usage and behavior, dynamic workloads, power management etc.

3.5.2.2 CloudAnalyst

CloudAnalyst is an extension of CloudSim that has been specifically designed for analyzing the cost and performance of geographically distributed data centers [175]. CloudAnalyst has been provided with a GUI, where users can drag and drop various components for building an extensive cloud system of his choice.

Hence experimenting with CloudAnalyst does not require an extensive programming knowledge. It also enables a researcher to carry out a series of simulation experiments with different parameters with relative ease.

3.5.2.3 Network CloudSim

Network CloudSim is an extension of CloudSim with scalable network and application model [176]. This enables users to evaluate scheduling and resource provisioning policies more accurately or optimizing the cloud system performance. Network CloudSim supports the modeling and study of generalized network applications such as distributed programming, high performance computing applications, workflows and e-commerce. Network CloudSim uses Network Topology class that implements the network layer in CloudSim, reads a BRITE file known as the topology file for generating a topological network. The topology file contains nodes, number of entities in the simulation that allows users to modify scale of simulation without changing the topology file.

3.5.3 Experiment Environment

This section presents the details of the experiment environment employed as part of the performance evaluation in this research. The description of experiments presented in Chapters Four, Five and Six were carried out using GNU Octave version 3.6.3 and CloudSim-3.0 on the Microsoft Windows 7 Professional Operating System installed with Java version 7. Every experiment was conducted for a fixed duration of 1000 seconds and the simulation runs were repeated 10 times and the mean value was considered to remove the temporary biases in the results [204]. The following sub sections explain the experimental environment employed in this research in detail

3.5.3.1 Experiment Steps

In order to carry out the experiments in a methodical manner and to confirm the repeatability of them, it is necessary to break or decompose the entire process into specific steps. It has been found that irrespective of the type of problem or objective of the study the process used for the simulation study is constant [205, 206, 207, 208]. The simulation process is generally divided into eight basic steps [209]. These steps are grouped into pre-software and software stages as shown in Figure 3.12.

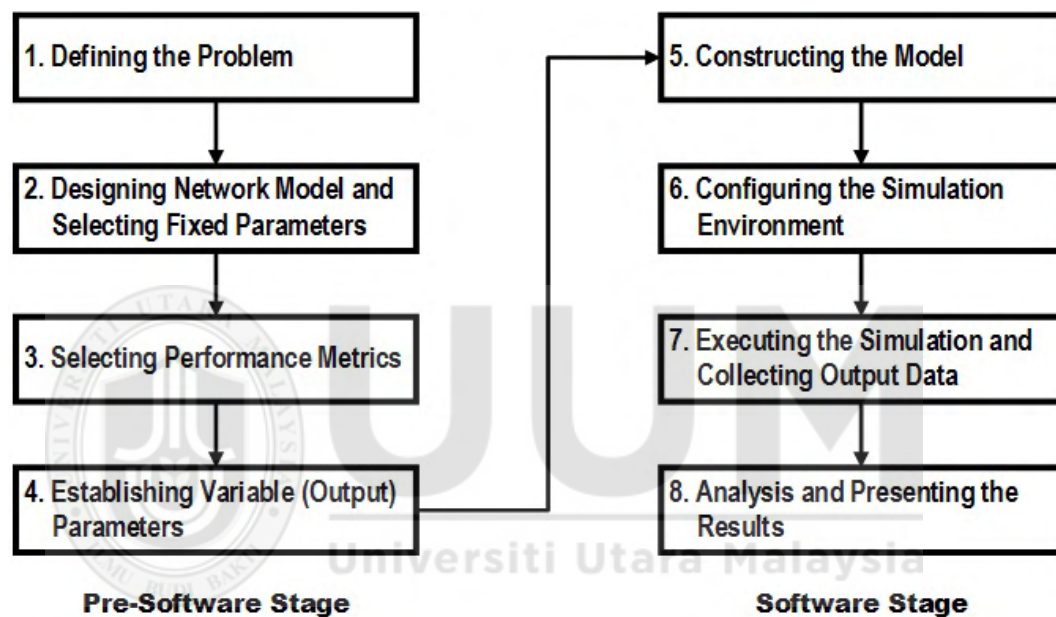


Figure 3.12. Simulation Steps (Adapted from [209])

The details of the steps are as follows:

- i. The initial step involves clearly identifying the research problem to be investigated along with defining the goals and objectives precisely. At this step, care must be taken to identifying and selecting the most appropriate simulation tool to be used.
- ii. Once the objectives are clear, the network topology to be simulated must be designed using paper-pencil method. The design should also include a suitable set of parameters reflecting the valid real world scenarios.

- iii. When the design is ready, the appropriate set of performance metrics must be identified for evaluating the performance of the network.
- iv. In order to compute the performance of the network, it is necessary to identify the right output parameters. In this step, the appropriate set of output (variable) parameters to be observed is established.
- v. At this step, the network topology designed in Step 2 is implemented on the simulation tool selected.
- vi. Following the topology implementation, it needs to be configured with right attributes for all the components to reflect the selected scenario to be tested.
- vii. Once everything is in order, the simulation program must be executed for the specified period and the output data for the selected variable (output) parameters must be collected. It is necessary to conduct multiple simulation runs to make sure that the results are free of bias.
- viii. Finally the required performance metrics need to be computed using the output data collected in Step 7 and presented in the most appropriate format such as tables, graphs or charts along with an interpretation of them.

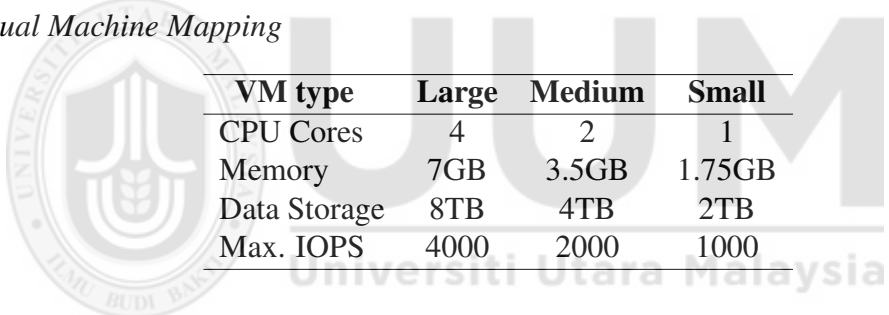
3.5.3.2 Experiment Setup

Data centers are the key components in cloud computing offering resources and services to customers [210]. Data centers host servers and other equipment necessary for providing a reliable service to customers. In the experiments conducted as part of this research also data centers take the center-stage. In the experiments many data centers were created with varying capacities and configurations to simulate a competitive environment similar to the real world. Each data center thus created were assumed to represent a different service provider (vendor). A federated cloud, where a vendor maintains multiple data centers that cooperate with each other to provide services to customers was taken into consideration in this research as it will complicate

the simulation process unnecessarily.

In order to achieve an experimental setup close to the real world environment, data centers with physical machines resembling public service providers were simulated. The configuration of some prominent public cloud service providers were obtained from literature and service providers' websites. A number of virtual machines of different types are mapped to physical machines depending on the configuration required. Three types of virtual machines such as small, medium and large as given in Table 3.3 were defined in order to create an environment to accommodate users with different requirements. This categorization has been based on the Microsoft Azure business cloud recommendations available at [211].

Table 3.3
Virtual Machine Mapping



VM type	Large	Medium	Small
CPU Cores	4	2	1
Memory	7GB	3.5GB	1.75GB
Data Storage	8TB	4TB	2TB
Max. IOPS	4000	2000	1000

A brief description on some of the key components used in simulation is given below:

Data center: is a collection of computers (servers) in homogeneous or heterogeneous configurations that are grouped and working together for handling customer requests.

Virtual machine: is a temporary allocation of processor power, memory, storage, and other resources based on a scheduling policy that can be used as if they are real. Virtual machines are allocated and removed on the fly on a user request. Multiple virtual machines can run on single hosts simultaneously and maintain processor sharing policies.

Host: consists of one or more CPU cores and fixed amount of memory and storage space that can be distributed to virtual machines based on a given allocation policy. Thus hosts can organize sufficient memory and bandwidth to the process elements to execute them inside virtual machines. Host is also responsible for creation and destruction of virtual machines.

Cloudlet: an application component (job) in the CloudSim environment that is responsible for delivering data in the cloud service model. Hence the length and output file size parameters of Cloudlets must be greater than or equal to 1. It contains various identities for data transfer and application hosting policy.

Table 3.4 shows the experiment setup attributes and the corresponding values.

Table 3.4
Experiment Setup Attributes and Values

Parameter	Sub-Parameter	Value
Response time		500 ms
Service time		10 minutes
Availability		100%
Computing Power	Processor	2.27 GHz Clock speed
		6 MB Cache
		533 MHz Bus speed
	Memory	2 GB
	Storage	100 GB
Network Speed		
Within data center	Bandwidth	100 Mbps
	Latency	3.33 ms
Between data center	Bandwidth	4 Mbps
No. of data centers	Quantification Mech.	1
	Trust Computing Mech.	1
	Trust Distribution Mech.	10
	Server per data center	1
No. of job submissions		10
Jobs per submission		1000
No. VMs per submission		1
No. cloudlets per job		1

3.5.3.3 Performance Metrics

Performance metrics is a set of attributes selected by researchers to measure and evaluate the operation of a specific equipment, algorithm, mechanism or protocol [212]. In order to assess the service quality and verify degree of fulfillment performance guarantees, it is necessary that the users are able to access and exchange performance metrics [213]. Thus the deployment of an appropriate monitoring infrastructure and collection of specific measurement data becomes vital in any public network including cloud computing systems. These performance metrics must represent the expectations and requirements of both customers and providers. Garg et al. in [32] have designed a set of Service Measurement Index (SMI) based on the International Organization for Standardization (ISO) standards for providing a standardized method for measuring and comparing public cloud services. The set of attributes developed by Garg et al. are Accountability, Agility, Assurance of Service, Cost, Performance, Security and Privacy and Usability [32]. Out of these broad attributes, the performance related attributes or metrics have been selected in this research for developing the service quality monitoring mechanism. The attributes that are frequently used for measuring service quality include response time, service time, accuracy, availability, integrity and reliability [32, 93, 110, 214]. Brief explanations along with the mathematical definitions of these metrics are as follows:

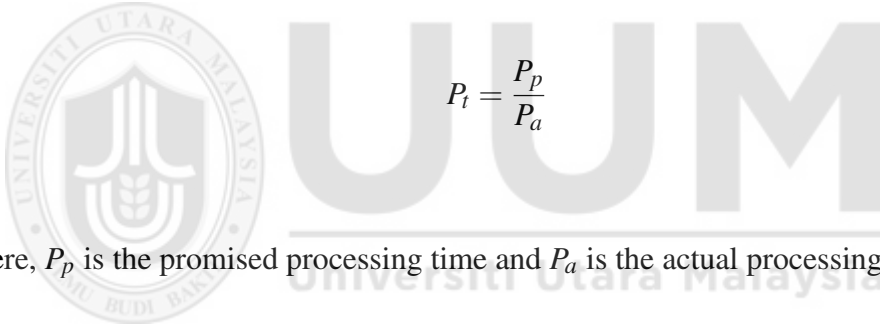
Response time: defined as the time elapsed between a customer's request and the cloud system's response [215, 210]. The apparent response time experienced by a customer would generally depend on many factors such as server delay in spawning a virtual machine, transmission delay and queuing delays at various points. Hence the absolute value of the response time can vary widely. So it is necessary to develop a normalized response time so that all the performance attributes are brought within the same range. So, a normalized response time parameter known as the Response time

efficiency R_t is defined by:

$$R_t = \frac{R_p}{R_a} \quad (3.1)$$

where, R_p is the promised response time and R_a is the actual response time.

Service time: is the total time taken for a process to complete from the time of submission to the time of delivery of the completed results [93]. The normalized processing time known as the Processing time efficiency P_t is defined by:



$$P_t = \frac{P_p}{P_a} \quad (3.2)$$

where, P_p is the promised processing time and P_a is the actual processing time.

Availability: is the proportion of times, a system is up and ready to accept and process jobs from clients immediately [214]. A system may become unable to accept job requests for many reasons including being shutdown for maintenance or upgrading, out of service due to failure of electricity or an external device such as networking equipment that prevents the customer data from reaching the system, an active attack on the system by a malicious intruder or being already overloaded [216]. Availability (A_v) can be expressed as a fraction given in the formula below:

$$A_v = \frac{A_k}{N_k} \quad (3.3)$$

where, A_k is the number jobs accepted for processing in a given in T and N_k is the total number of jobs submitted over the same time.

3.5.3.4 Confidence Level of Simulation Results

The performance evaluation needs to be repeated several times in order to remove the temporary biases in the results and to confirm the repeatability of them. Due to the randomness of the operations within the simulation environment, the results would also exhibit some random behavior within specific limits. Hence it is practically impossible to obtain exact values for the performance metrics calculated using these results. Therefore, mean values (μ) of the experiment results are generally computed. The values computed this way are commonly known as estimates and required to have some kind of confidence level associated with them [217]. In this research, every experiment was repeated 10 times and the statistical parameters were computed with 95 percent confidence level except for where it is specifically stated. The detailed steps of computing statistics from random samples are given in [218].

3.6 Summary

This chapter presented on the research methodology adopted in conducting this research. It consist of four phases namely, analysis, design, testing and evaluation. In order to arrive at a scientifically valid and practically proven methodology, the stages of the design research methodology haven been adapted and integrated into the different phases of this methodology. It has been proposed to develop different mechanisms for modeling quantifying and distributing of service quality in cloud computing. The activities that will be carried out and the outcomes expected at the end of reaching each milestone were highlighted in this chapter in the light of different phases of the methodology and the stages of DRM. Phase 1 (Analysis) that includes both Research Clarification and Descriptive Study-I stages of DRM focuses

on obtaining a clear understanding of the research. The main outcomes of this phase include identification of research problem, objectives and research questions, and development of a conceptual model of the research. The design and testing phases of the methodology concentrates on how to develop and test the mechanisms for their functionality. It has been highlighted the models developed to represent the mechanisms will be tested and functionality validated. The functional testing will be carried out using GNU Octave, an open source Matlab clone software. The last activity in the proposed research is the evaluation of the mechanisms. This activity will be carried out through extensive simulation with CloudSim, the most popular cloud computing simulator used by the research community in this area today.



CHAPTER FOUR

SERVICE QUALITY MODELING MECHANISM FOR CLOUD COMPUTING

4.1 Introduction

This chapter presents the proposed service quality modeling mechanism for cloud computing based on the research methodology established in Chapter Three. The proposed mechanism creates a single rating value known as the trust score using the performance metrics discussed in Chapter Three. As the initial step at developing the mechanism, the performance metrics were studied further detail with the aim of arriving at an objective criterion for evaluating the performance of cloud service providers based on these metrics. It has also been emphasized that the mechanism developed as part of this research must be extensible to include other parameters as well for meeting the requirements of a wide range of customers.

The organization of this chapter is as follows. Section 4.1 provides an introduction to the chapter highlighting the importance of a single rating value for identifying the service quality of cloud providers. Section 4.2 explains how the parameters are normalized, so that all the parameter values are restricted to a single range between 0 and 1 included. Section 4.3 describes how the service quality of cloud service providers can be quantified using the normalized parameters explained in Section 4.2. Sections 4.4 and 4.5 present the Multi-Parameter Service Quality Quantification Mechanism, one of the main contributions of this research and its functional evaluation respectively. Finally Section 4.6 concludes the chapter providing a brief summary of the complete chapter.

4.2 Normalizing of Performance Metrics

The performance metrics defined in Chapter Two show different behaviors depending on the type of parameter. Some parameters such as response time and service times are continuous variables starting from 0 and grow to very large values and measured in seconds or fractions of it. On the other hand, availability, reliability and integrity are probability values which vary between 0 and 1 continuously. Also the response and service time must be computed with respect to the required times and the decision must be arrived as whether the service provider exceeds, meets or fails to meet the expectation of the client. Thus these values must be positive, zero or negative depending on the performance.

Since each parameter has a different behavior and range, it is not possible to compare or combine parameters for the purpose of arriving at single objective metric that can be used for ranking service providers. Thus, bringing them within a single range is very important as more than one parameter, so that they all can be treated similarly when qualifying the service quality of different service providers based on one of more parameters. Bringing the parameter values to a single range is known as normalization of parameter values. The performance parameters discussed in Subsection 2.3.3 have been normalized as follows:

Response Time Factor (R_t): the normalized response time known as response time factor (R_t) is computed as given by Equation 4.1:

$$R_t = \begin{cases} 0 & : R_a \geq 2R_p \\ \frac{(2R_p - R_a)}{2R_p} & : 0 \leq R_a < 2R_p \end{cases} \quad (4.1)$$

where, R_p is the required response time and R_a is the actual response time.

From Equation 4.1, it can be seen that the value of R_t varies between 0 and +1. The negative extreme value of R_t has been confined to 0 in order to limit the values of response time factor within a specified range.

Service Time Factor (S_t): the normalized service time known as service time factor (S_t) is computed as given by Equation 4.2:

$$S_t = \begin{cases} 0 & : S_a \geq 2S_p \\ \frac{(2S_p - S_a)}{2S_p} & : 0 \leq S_a < 2S_p \end{cases} \quad (4.2)$$

where, S_p is the required service time and S_a is the actual service time.

It is clear Equation 4.2 that the value of S_t varies between 0 and +1. The negative extreme value of S_t has been confined to 0 in order to limit the values of service time factor given a specified range.

Availability Factor (A_v): the normalized availability known as the availability factor (A_v) can be expressed as a fraction as given in Equation 4.3:

$$A_v = \frac{1}{2} \left(\frac{A_k - R_k}{N_k} + 1 \right) \quad (4.3)$$

where, A_k and R_k is the number jobs accepted and refused for processing in a given in T while N_k is the total number of jobs submitted over the same time.

Also $R_k = (N_k - A_k)$

Thus Equation 4.3 can be simplified as:

$$A_v = \frac{1}{2} \left(\frac{2A_k - N_k}{N_k} + 1 \right) \quad (4.4)$$

Equations 4.3 and 4.4 take both the positive and negative responses of the in computing the availability factor. This is slightly different from the generic availability rate computed from only positive responses [104]. The advantage of the availability factor introduced in this thesis is; it penalizes the misbehaving service providers who continue to perform below the expectations of the customers. The factor given by Equations 4.3 and 4.4 can also differentiate between a new service provider and a one who always perform below the expectations. The value of A_k ranges between 0 and +1, with 0.5 as the origin representing the newest system.

Reliability Factor (R_e): the normalized reliability factor (R_e) of the system is defined as in Equation 4.5 :

$$R_e = \frac{1}{2} \left(\frac{2C_j - N_j}{N_j} + 1 \right) \quad (4.5)$$

where, C_j is the number jobs successfully completed in a given within time T and N_j is the total number of jobs accepted for processing over the same time.

The range of R_e is between 0 and +1 with 0.5 being the value given to a new service

provider.

Integrity Factor (I_f): the normalized integrity factor I_f is given by Equation 4.6 :

$$I_f = \frac{1}{2} \left(\frac{2J_o - J_t}{J_t} + 1 \right) \quad (4.6)$$

where, J_o is the number jobs completed preserving operational integrity in a given in T and J_t is the total number of jobs processed over the same time.

Similar to other factors, the value of I_f also lies between 0 and +1 with 0.5 assigned to the service provider who just started offering his services.

It could be seen that the normalized performance factors defined above follow similar pattern in computing the performance of the cloud computing based on different metric. Also all the factors now behave like probabilities ranging between 0 and +1 included while them being random events. Hence probability theories can be easily adopted in dealing with these factors. The same principle used for converting the selected performance metrics into performance factors can be easily extended without much difficulty to cover other performance metrics to cater to any user requirement or future demand.

4.3 Modeling of Service Quality of Cloud Providers

Normalized performance metrics (performance factors) defined in 4.2 provide an objective measure (trust score) for characterizing the performance of cloud service providers. Although these factors could be used as an objective measure for comparing different service providers based on their performance, this comparison will be limited

to a single parameter only. Also, there is another shortcoming in the above said comparison. All but response time and the service time provide a direct probability value of the performance of the service provider based on the selected parameter. Hence they can be directly used to compare and rank the service providers. On the other hand, the expectations of clients on their response times and service times may differ from one customer to another. Hence, it is not practical to compute the response or service time factors based on a single fixed value for these. Thus, Equations 4.1 and 4.2 must be applied independently to each customer separately.

4.3.1 Single Parameter Service Quality Quantification Mechanism (SP-SQQM)

In order to verify the functionality of the mechanism developed in Section 4.3, a computer model was developed. The algorithm for single parameter service quality quantification mechanism using response time as an example is given in Algorithm 4.1.

Algorithm 4.1: Single Parameter Service Quality Quantification Algorithm

```

required response time =  $\tau_r$ 
actual response time =  $\tau_a$ 
compute normalized parameter ( $\delta$ ) =  $\frac{|\tau_r - \tau_a|}{\tau_r}$ 
if ( $\tau_a \leq \tau_r$ ) then
  if ( $T_n == 1$ ) then
     $T_{n+1} = T_n$ 
  else
     $T_{n+1} = T_n + \delta * T_n : T_0 = a \text{ and } n = 1, 2, \dots$ 
  end if
else
  if ( $T_n == 0$ ) then
     $T_{n+1} = T_n$ 
  else
     $T_{n+1} = T_n - \delta * T_n : T_0 = a \text{ and } n = 1, 2, \dots$ 
  end if
end if

```

where, a – initial trust value

This algorithm was implemented using GNU Octave and executed under different

conditions to verify the behavior of the mechanism. A simple cloud environment with one server and two clients was setup in order to carry out this experiment. The server was configured in such a manner that it has a fixed response time of 500 ms for both clients. The clients were configured to send requests at the regular intervals with the expected response times of 300 ms and 700 ms respectively. The data for each client was collected separately as they need to be analyzed independently. Figure 4.1 shows the change in trust score due to continuously the performance of the system is better or worse than the expected one.

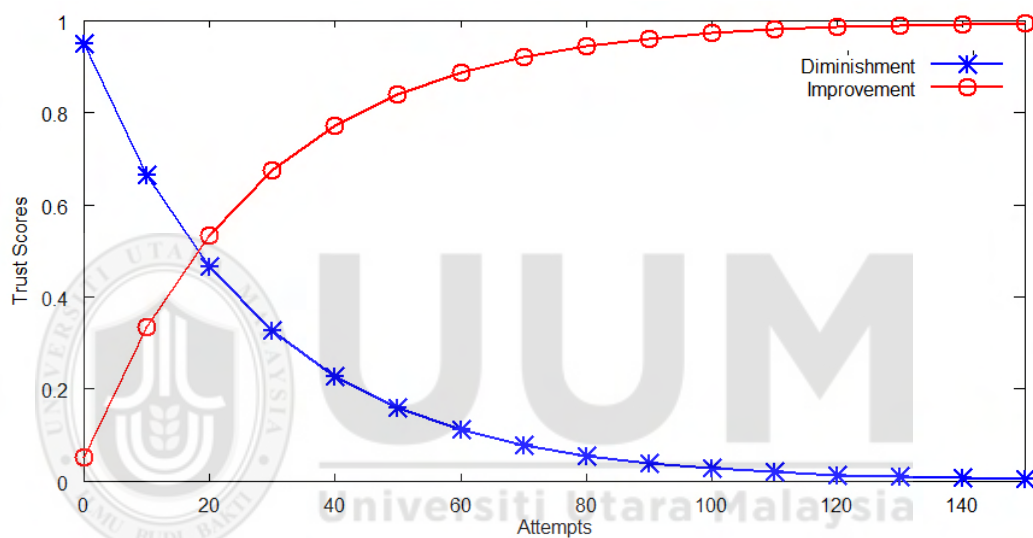


Figure 4.1. Change in Trust Scores

From Figure 4.1, it can be seen that in both cases the trust scores were either monotonously improving or diminishing due to continuous responses of the same kind. Also, it could be observed that during both improvement as well as decrement phases, the change is initially faster and then slowly and asymptotically approach the end values such as perfect trust and no trust represented by +1 and 0 respectively.

The experiment was modified to verify the behavior of the mechanism under competing demands from customers. The modified experiment environment was designed with a single server capable of spawning four virtual machines to serve four different customers with different response time requirements. The response

requirements were set up 100ms, 400ms, 700ms and 900ms respectively. These are shown as 0.1, 0.4, 0.7 and 0.9 time units for convenience. The actual response time of the cloud server was set to a range between 300ms and 500 ms. Figure 4.2 shows the change in trust scores experienced by customers with different performance expectations.

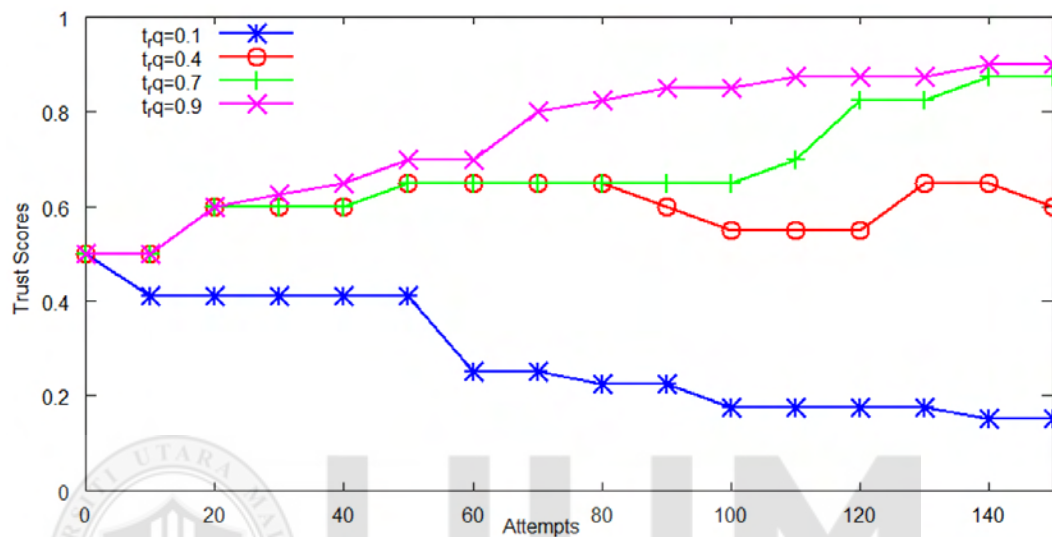


Figure 4.2. Comparative Change in Trust Scores

From Figure 4.2, it can be seen that whenever a more stringent requirement has been met, all the trust values of the relaxed requirements have also been improved. This is due to the reason that, if the system could meet a stringent condition it could easily meet relaxed requirements. This fact should reflect on the trust scores and hence all the respective trust values have been positively updated. Conversely, when a relaxed condition is not met, all the trust scores of the more stringent requirements are reduced. This is due to the reason that the failure to meet a relaxed requirement would necessarily an indication that more stringent performance requirements will not be met.

Figure 4.2 also shows that the most stringent condition indicated by the requirement of 0.1 (100ms) continues to decline. This is due to the reason that the negative performance of the system for any requirement lower than this requirement would

affect this one. Hence, it is obvious from the results that it is very difficult to meet strict performance requirements unless special attention has been paid to these requirements. On the other hand, the trust score of the most relaxed performance requirement designated by the response times of 0.7 (700ms) and 0.9 (900ms) show continuous improvement. This is due to the collective improvement of all the more stringent requirements. The other plot belonging to the customer requirement of 0.4 (400ms) show mixed results of trust score going up and down based on the actual response time varying around the required response time.

4.4 Multi-Parameter Service Quality Quantification Mechanism (MP-SQQM)

Sub Section 4.3.1 presented the single parameter service quality quantification mechanism that can quantify and rank the cloud computing providers based on the performance. The shortcoming of the SP-SQQM has already been highlighted as it depends only on one selected parameter. According to Carrera et al. [92] real-time applications demand better response times and throughput as opposed to non real-time batch jobs that are more concerned with accuracy and lower processing times. Hence customers and applications will not be satisfied with single parameter based ranking of service providers. In order to meet this requirement, it is necessary to have a mechanism that can accept multiple parameters as inputs and compute the single output score.

In this section, the SP-SQQM is extended to accommodate multiple parameters for producing a trust score that can be used to compare and rank service providers. The proposed mechanism is built on the probability of the expected outcomes represented by a set of performance metrics. Bayes' rule links the prior probabilities and conditional probabilities to the posterior probability of an event [219].

Bayes' rule for two related events may be expressed as [220]:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \quad (4.7)$$

where: A and B are two related events.

$P(A)$ and $P(B)$ are the probabilities of A and B occurring independently.

$P(A|B)$ and $P(B|A)$ are conditional probabilities that the probability of A provided B is true and the probability of B given A has already occurred respectively.

The Bayes Networks have been commonly used to represent probabilistic relationship between causes and effects [221]. A Bayesian network is represented as a directed acyclic graph with nodes representing variables and edges representing the conditional dependencies. A conditional probability table giving the probability value of the random variable represented by the node conditional on the parent is associated with each node. The theoretical foundation for all the Bayesian networks is derived from the Bayes' rule [222].

A Naive Bayesian Network is a simple but powerful Bayesian network that assumes that each parameter is independent of each other provided it has the class variable as its unique parent [223, 224]. The advantages of Naive Bayesian Network over other networks include its known structure and efficient classification process [225]. Both these advantages are the result of the assumption that all the features are independent of each other. Figure 4.3 shows the structure of a Naive Bayesian Network with the output

(Class) parameter at the top and the input parameters at the leaf nodes. The conditional probabilities associated with each parameter would be given in a performance table.

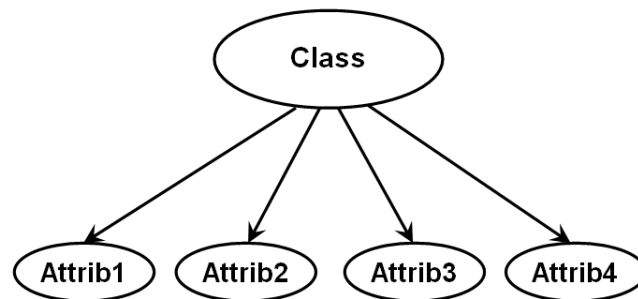


Figure 4.3. Naive Bayesian Network

In order to compute the trust score of a cloud computing system, it is necessary to represent the performance metrics in a Naive Bayes Network similar to Figure 4.3. In such a network, the output node would represent the trust to be computed while the leaf nodes would be the service quality metrics. The conditional probabilities associated with the metrics will be used to create the performance table. Figure 4.4 and Table 4.1 show the directed graph and the associated performance table created for a generic condition for a given cloud computing system respectively.

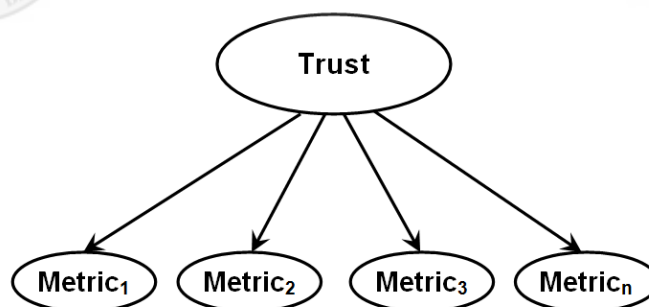


Figure 4.4. Naive Bayesian Network for a Cloud Computing System

where:

$O = 1$ represents the successful outcomes and $O = 0$ represents unsuccessful outcomes.

Table 4.1
Performance Table

Metric	No. of interactions with O=1	No. of interactions with O=0
$Metric_1$	s_1	n_1
$Metric_2$	s_2	n_2
$Metric_3$	s_3	n_3
$Metric_n$	s_n	n_n

s_i and n_i represents the number of times the $Metric_i$ is considered to have met its requirements when the overall outcome is successful and unsuccessful respectively.

From Table 4.1, it is possible compute all the required conditional and absolute probabilities.

The trust score for the given cloud computing system for the given set of performance metrics is given the joint probability in Equation 4.8.

$$T = p(O \cap m_1 \cap m_2 \cap \dots \cap m_n) \quad (4.8)$$

where O and m_i represent the outcome and i^{th} metric, when all of them met and final outcome is considered successful. The joint probability can be represented as a conditional probability and simplified as given below using Bayes' theorem.

$$T = p(O)(m_1, m_2, \dots m_n | O)$$

and

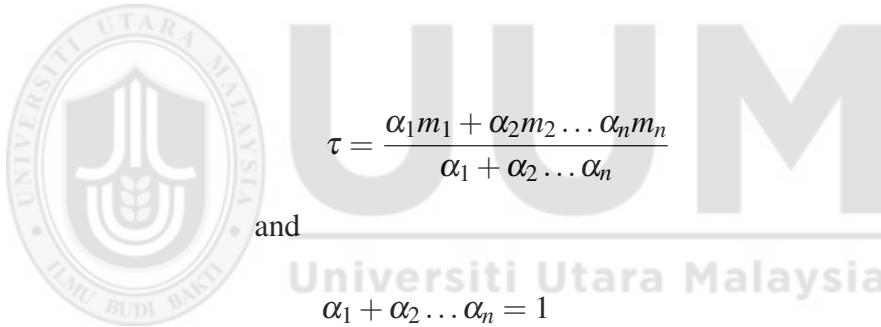
$$T = p(O)p(m_1|O)p(m_2|O) \dots p(m_n|O) \quad (4.9)$$

provided the metrics (m_i) are independent of each other.

Table 4.1 will be continuously updated by the trust provider, whenever a client obtains the services from a given cloud computing node. Hence it must maintain a performance table similar to Table 4.1 for each cloud computing node and continuously update it. Whenever a client request the trust value of a node, it is possible to compute it using Equation 4.9 based on any set of performance metrics provided by the client.

4.4.1 Computing Trust Score with Different Priorities

It is also possible to compute the trust value using Equation 4.9, when a client has different priorities for different metrics. When such a prioritization of metrics is required, it will be given by metric weights as shown in Equation 4.10.



$$\tau = \frac{\alpha_1 m_1 + \alpha_2 m_2 \dots \alpha_n m_n}{\alpha_1 + \alpha_2 \dots \alpha_n} \quad (4.10)$$

and

$$\alpha_1 + \alpha_2 \dots \alpha_n = 1$$

where m_i is the i^{th} parameter and α_i is the weight applied to it.

For example, if a client who seeks a certain service with service quality requirements on metrics m_1 , m_3 and m_7 with the priorities of α_1 , α_3 and α_7 , then the performance table would be modified as shown in Table 4.2.

The values in the modified columns will be used for computing the conditional probabilities and the trust score for the node.

Table 4.2
Modified Performance Table

Metric	Weight	No. of interactions with O=1		No. of interactions with O=0	
		Original	Modified	Original	Modified
m_1	α_1	s_1	$\alpha_1 s_1$	n_1	$\alpha_1 n_1$
m_3	α_3	s_3	$\alpha_3 s_3$	n_3	$\alpha_3 n_3$
m_7	α_7	s_7	$\alpha_7 s_7$	n_7	$\alpha_7 n_7$

4.5 Functional Verification of MP-SQQM

The verification of the functionality of MP-SQQM was carried out using a computer model developed using GNU Octave software two parameters, response time and processing time. The verification was limited to two parameters as more than independent variables cannot be plotted on a three dimensional graph and hence would be difficult to visualize the performance. If the functionality can be verified for two input parameters, the mechanism can be extended to include any number of parameters as explained in Section 4.4. The pseudocode for the two parameter MP-SQQM is given in Algorithm 4.2.

where:

S_R and S_P represent the number of times the customer requirements were met successfully and N_R and N_P represent the number of times the system failed to meet the customer requirements respectively.

The algorithm was tested extensively for functionality under different conditions. Initially the experiment was carried for the condition when a user requires both the conditions to be met. That is the user has equal priority for both parameters and hence the weight is set to be $\alpha_1 = \alpha_2 = \frac{1}{2}$. Figure 4.5 show the variation in trust scores under the above specified condition.

Figure 4.6 shows the effect of different weights on the final score. It can be observed

Algorithm 4.2: Multi Parameter Service Quality Quantification Algorithm

required response time = R_r

actual response time = R_a

required processing time = P_r

actual processing time = P_a

if ($R_a \leq R_r$) **then**

$S_R = S_R + 1$

else

$N_R = N_R + 1$

end if

if ($P_a \leq P_r$) **then**

$S_P = S_P + 1$

else

$N_P = N_P + 1$

end if

APPLY weighs α_1 and α_2

COMPUTE the joint probability (T) using individual probabilities

Trust Score = T

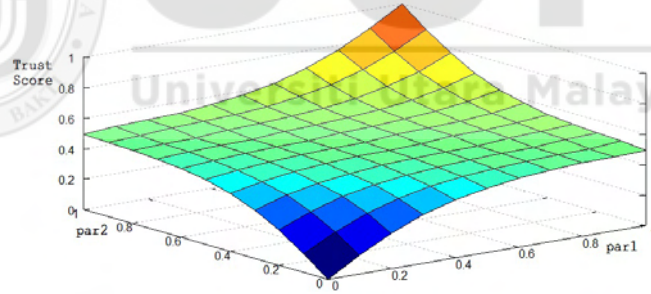


Figure 4.5. Trust Score Computed Using Two Input Parameters

from Figure 4.6 the final score computed is same irrespective of the scale that has been used to specify the weights. That is when the equal weights are specked as $\alpha_1 = \alpha_2 = 1$ or $\alpha_1 = \alpha_2 = \frac{1}{2}$, the scores computed are same. This is due to the fact that the formula itself computes the fractional weight while applying them to the probability values.

The effect of the weights applied on the individual trust scores on the final (or combined) trust score were verified through another set of experiments. In these

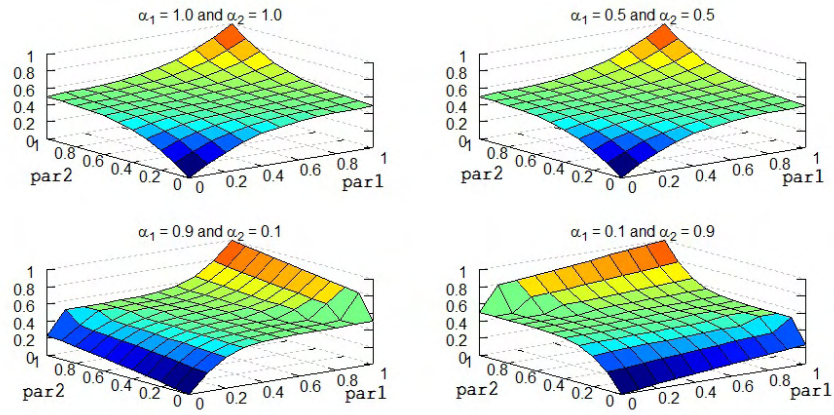


Figure 4.6. Effect of Weights on Trust Scores Computed

experiments, the individual trust scores were fixed at $t_1 = 0.7$ and $t_2 = 0.5$ and then the weight w_1 applied to one input score namely t_1 were changed progressively from 0 to 1 while maintaining the relationship w_2 fixed at 0, 0.2, 0.4 and 0.5. The results are shown in Figure 4.7 shows effect of weights on the final trust score clearly. The value of the final trust score lies between the individual trust scores depending on the weights applied. The effect of one input parameter on the final trust score can be totally eliminated by applying a weight ($w = 0$). This fact is illustrated by the blue line shown to lie at 0.7 from the beginning to end. When the effect of one parameter is eliminated, the final (combined) trust score takes the value of the other parameter automatically.

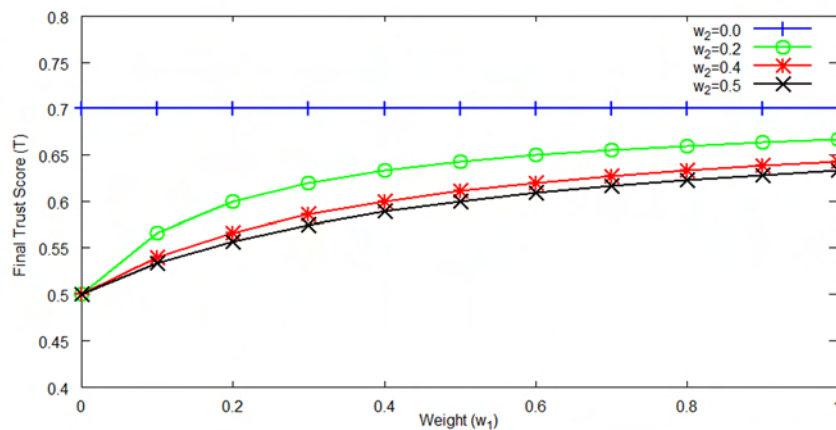


Figure 4.7. Effect of Weights on Final Trust Score

Figures 4.5, 4.6 and 4.7 show that the multiple input parameters can be successfully

combined into a single score for quantifying the service quality of cloud service providers. Further, by applying weights to different service quality parameter, the final trust score can be modified to reflect the requirements of different groups of users. From Figure 4.7, it can be seen that the same cloud system with the same performance may show different capabilities to different groups of users depending on their requirements reflected by the parameter weights. Thus the MP-SQQM performs correctly as expected combining multiple parameters to a single trust score based on user requirements.

4.6 Summary

This chapter presented the two service quality modeling mechanisms for cloud computing developed as part of this research. The service quality modeling mechanism were named Single Parameter Service Quality Quantification Mechanism (SP-SQQM) and Multiple Parameter Service Quality Quantification Mechanism (MP-SQQM) depending the number of inputs used. The proposed mechanisms were tested for functionality under limited laboratory conditions using simulations. The simulation results show that both mechanisms are functioning as expected. The performance analysis of the mechanisms will be presented in Chapter Seven.

CHAPTER FIVE

ADAPTIVE TRUST COMPUTING MECHANISM FOR CLOUD COMPUTING

5.1 Introduction

Chapter Four identified and presented a set of metrics that can be used in quantifying the performance of cloud services. Further to presenting the metrics, Chapter Four presented two specific algorithms that can be used for quantification of performance of cloud services based on those metrics. These algorithms accept performance attribute values as inputs and convert them to a single score that can be used to compare and rank different cloud providers. But the scores thus computed depend only on the single performance values just entered and do not reflect the past performance of the system. Hence the score is not really representative of the performance of the system. In order to overcome this limitation, Chapter Five presents the adaptive trust computing mechanism that can closely track the performance and compute a score based on its past performance under similar requirements.

The organization of this chapter is as follows: Section 5.1 provides a brief introduction of the chapter highlighting the importance of tracking performance of the system and computing the trust score based on its long term process rather than using a single instance. Section 5.2 presents the principles of trust formation at the inception and evolution based on the performance of the system onwards. Sections 5.3 to 5.5 present the preliminary mechanisms developed in this research. Sections 5.6 and 5.7 present the main contributions of this chapter, namely the Robust Adaptive Trust Computing Mechanism (RATComM) and Multi-Dimensional Trust Computing Mechanism (MuDTComM). Both mechanisms combine the important features of the algorithms described from Sections 5.3 to 5.5. Finally Section 5.8 concludes the

chapter by summarizing the main contributions of the chapter.

5.2 Trust Formation and Evolution

The management of trust is the process that goes through several distinct stages either continuously improving or diminishing based on the experience of an individual [226]. During the life cycle of trust, it is possible to identify three distinct functions playing an important role in trust management [227]. These functions are trust formation, trust evolution and trust distribution. Trust formation is the initial idea formed of a new object in terms of its trust worthiness. With respect to computing systems or service providers who are new to the market will be required to create an initial impression on the customers in terms of their capabilities. Hence the initial trust score given to a new service provider can either take a neutral value or computed using the published capacities of the resources. Taking the neutral score would be safer and easier as this initial score would immediately be modified once the system starts interacting with users. In this research, the initial score for any service provider is taken as the neutral value (0.5) between the extreme values of most trustworthy (1) and least trustworthy (0).

With time and more interactions with the other party, the value of trust changes either positively or negatively becoming stronger or weaker respectively [228]. This is commonly known as trust evolution. An adaptive trust evolution mechanism must modify the trust scores continuously based on the performance of the system. When the system meets or exceeds the performance demanded by the clients, the trust score must be improved. On the other hand, if the system fails to meet the requirements, the trust scores must be diminished accordingly.

This chapter presents several trust evolution algorithms developed as part of this

research. The adaptive trust computing mechanism presented towards the end of the chapter is developed by taking a positive aspects of all the algorithms presented priorly. Trust distribution is the exchange of trust information between cooperating nodes (trust computing units) about a single service provider. This topic is discussed in detail in Chapter Six.

5.3 Adaptive Continuous Trust Evolution Mechanism (ACTEM)

Public cloud computing systems are accessed by users with varying requirements [229]. Some of the customers may be very strict on their requirements while on the other extreme, some customers might tolerate a little more deviation from the committed performance guarantees. Hence, based on general requirements, customers can be grouped into three main categories. They are namely;

Type I - Customers who require a guaranteed level of service

Type II - Customers who require an average level of service

Type III - Customers who require basic level of service with no guarantees

The requirements of Type I customers are more stringent than Type II and III customers as they demand a guarantee on the commitments of service quality. It is impractical to provide an absolute guarantee that the required service quality will always be met, unless dedicated systems with ample resources are permanently allocated [230]. In such situations, the most practical way is to provide a statistical guarantee within a specified confidence level. Depending on the stringency of the requirements, users can demand different service levels specified by specific confidence levels such as 90%, 95% etc. The charging scheme can also be adapted to suit the service levels, where users are required to pay premium prices, if more stringent service quality is demanded. This kind of differentiated charging scheme

can be easily justified, as the service providers are to reserve more resources to meet the requirements of these customers. Hence the trust evolution mechanism must be capable of distinguishing between different performance demands for the same performance value.

In order to continuously modify the trust score based on the performance, it is necessary to monitor the performance of the system in real time. We propose to have a feedback mechanism that will continuously monitor and track the system performance and supply the performance values to the trust evolution mechanism. Figure 5.1 shows the block diagram of a trust management system comprising service monitor, trust formulation and trust evolution units.

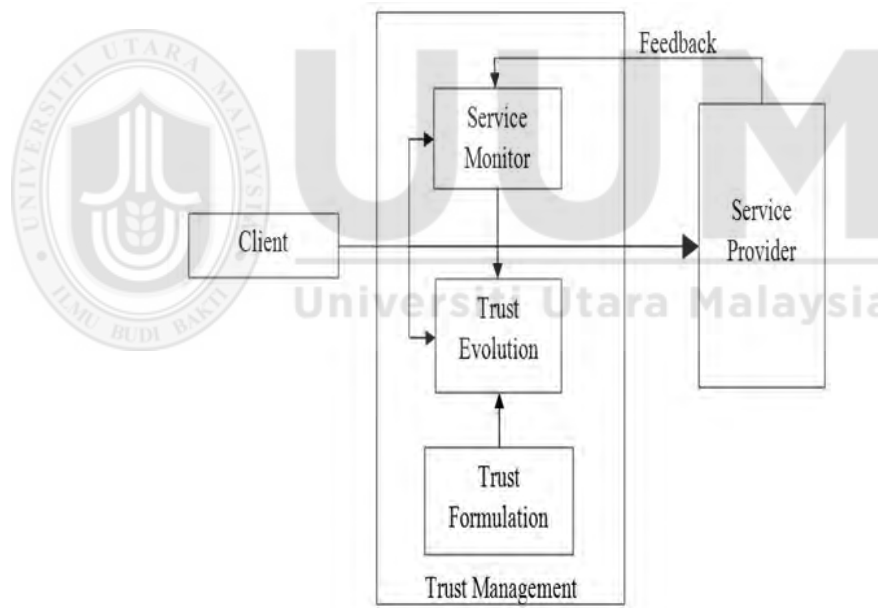


Figure 5.1. Trust Management System

The operation of the trust management system is as follows: the service monitor receives its inputs from both the clients and the service provider. The clients send both the required performance value for a given metric and the confidence interval. The actual performance value is obtained from the service provider. The trust evolution module computes the new trust score and updates the trust value based on the inputs

received. The Adaptive Continuous Trust Evolution Mechanism included in the trust evolution module is given in Algorithm 5.1 using response time as the performance metric.

Algorithm 5.1: Adaptive Continuous Trust Evolution Algorithm

required response time = τ_r

actual response time = τ_a

confidence level = 95%, 90% etc.,

compute the mean response time ($\bar{\tau}$) = $\frac{\sum_{i=1}^N (\tau_a)_i}{N}$

compute confidence interval for mean response time = $\bar{\tau} \pm t_{(\alpha/2, N-1)} s / \sqrt{N}$

compute normalized parameter (δ) = $\frac{|\tau_r - \bar{\tau}|}{\tau_r}$

if ($\bar{\tau} - t_{(\alpha/2, N-1)} s / \sqrt{N} \leq \tau_r \leq \bar{\tau} + t_{(\alpha/2, N-1)} s / \sqrt{N}$) **then**

if ($T_n == 1$) **then**

$T_{n+1} = T_n$

else

$T_{n+1} = (1 - \delta) * T_n + \delta : T_0 = a \text{ and } n = 1, 2, \dots$

end if

else

if ($T_n == 0$) **then**

$T_{n+1} = T_n$

else

$T_{n+1} = (1 - \delta) * T_n - \delta : T_0 = a \text{ and } n = 1, 2, \dots$

end if

end if

where, a – initial trust value, N – No. of results in the buffer

5.3.1 Functional Verification of ACTEM

The proposed algorithm was functionally verified in a simulated environment. The computer model for simulation was created using GNU Octave with a trust management system that consisted of service monitor and trust evolution unit. The simulation was limited to single performance metric, response time. In order to simplify the tests, the required response time was fixed, while the actual response time was allowed vary within two extreme values. Figure 5.2 shows the change in trust score during continuous positive or negative feedbacks.

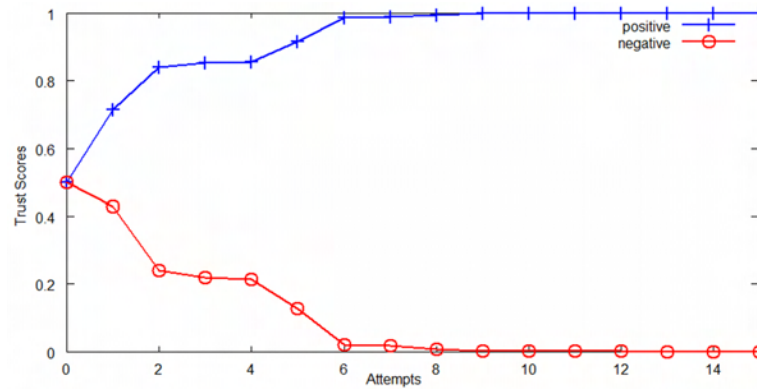


Figure 5.2. Trust Scores due to Continuous Positive or Negative Feedbacks

From Figure 5.2, it can be seen that the continuous reception of positive feedbacks improves the trust scores towards the positive extreme 1 while the continuous negative feedbacks diminishes the trust score towards 0 starting from the neutral value of 0.5.

In order to test the effect of confidence level on the trust score, the experiment was slightly modified. The number of clients was increased to two one expecting the response time to be within the expected limit with a confidence level of 90%, while the other one to be a little bit more stringent with a confidence level of 95%. The experiment was repeated several times, by fixing the actual response time to fall between the two demands. Figure 5.3 shows the effect of confidence level on the trust scores despite the response time being the same.

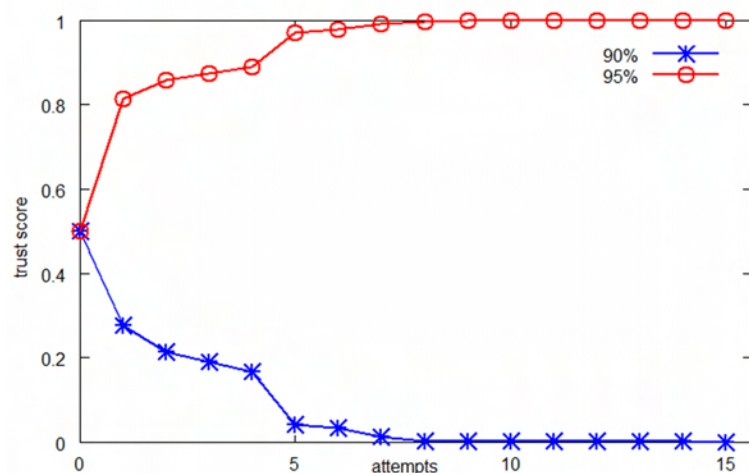


Figure 5.3. Effect of Confidence Level on Trust Scores Computed

Figure 5.3 shows the effect of the stringency of the client demand on the trust placed on the service provider. Stronger the stringency, less the trust would be for the same performance of the system. On the other hand, a less demanding client can be satisfied with slightly lower performance that is required for a more demanding customer. Thus, it can be concluded that the ACTEM can be used to check for the performance of a cloud system by any client irrespective of his demand level by providing the required parameter values.

5.4 Memoryless Trust Computing Mechanism (MemTrust)

Similar to ACTEM, MemTrust also receives the inputs from both the cloud system and the clients based on which the trust scores for the system would be computed. The inputs received from the client and the system are the required and the actual performance values for any given metric respectively. The distinguishing feature of the MemTrust is that the computed trust score does not depend on the past performance of the system but the recent performance of the system and the shape of the Sigmoid function used. Figure 5.4 shows the functional level detailed diagram of the MemTrust trust evolution unit. The main functional components that make the trust evolution unit include the performance value storage block, the summing point and the Sigmoid function.

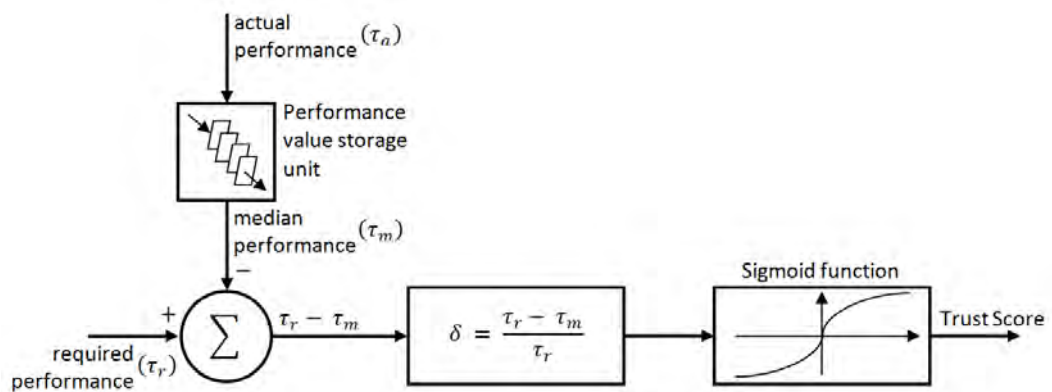


Figure 5.4. MemTrust Trust Evolution Unit

The operation of the trust evolution unit is as follows: the performance values for a specific metric received by the monitoring unit are stored in the storage block on a First In First Out (FIFO) basis. Depending on the capacity of the storage block, the number of most recent performance values to be stored is predetermined and fixed. From these stored values, the median performance value is computed. By taking the median of the performance parameter eliminates the effects of momentary fluctuations and provides somewhat a stable outlook for the system. Median has been selected as the preferred statistic opposed to mean as it is least affected by outliers, though both are commonly used to compute the central tendency of a particular attribute [231]. The summer computes the difference between the required performance value and the median performance value and then passes the result to the next stage. At the next stage, the normalized difference (δ) in the required performance and actual performance is computed and forwarded to the Sigmoid function as the input. The Sigmoid function then computes the trust score based on this input. Since the Sigmoid function directly computes the trust score using the normalized difference, it is not necessary to save the past trust scores computed. Hence this method does not have the memory of any previous trust scores. It is only necessary to track the actual response times of the cloud provider in order to compute the normalization parameter (δ).

The Sigmoid function is used in this algorithm due to its special properties that are suitable for this kind of operation. The special properties that can be useful for trust management are [232]:

- Shape of the curve (S-shape)
- Linear operation in the middle region
- Large input range (from $-\infty$ to $+\infty$)

Figure 5.5 shows the shape of a Sigmoid curve. The nonlinear property of the Sigmoid function makes it very difficult to push the scores towards extreme values both positive and negative. The linear portion of the curve can be used to amplify the trust score differences in that region as most of the customers could be expected to operate there. The large input range makes the system robust as theoretically it would be impossible to break the system using extreme values.

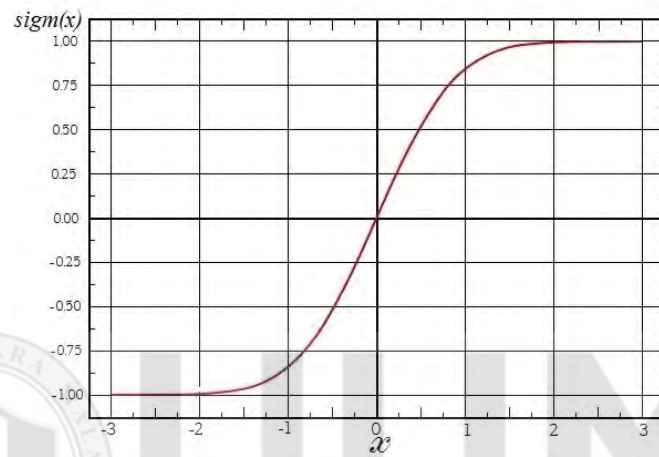


Figure 5.5. Sigmoid Function

The Sigmoid function needed to be modified to change its range from $(-1, +1)$ to $(0, +1)$. This modification is required as the trust scores in this research has been specifically restricted to be within the extreme values of 0 and +1. Figure 5.6 shows the shape of the modified Sigmoid function used in this research.

Algorithm 5.2 lists the pseudo code of the algorithm used to implement the MemTrust.

Algorithm 5.2: MemTrust: Memoryless Trust Evolution Algorithm

required response time = τ_r
actual response time = τ_a
compute the median response time = τ_m
compute the difference in response time $\tau_d = (\tau_r - \tau_m)$
compute the normalized response time $\delta = \frac{\tau_d}{\tau_r}$
compute the trust score using the Sigmoid function $(T) = \text{Sigm}(\delta)$

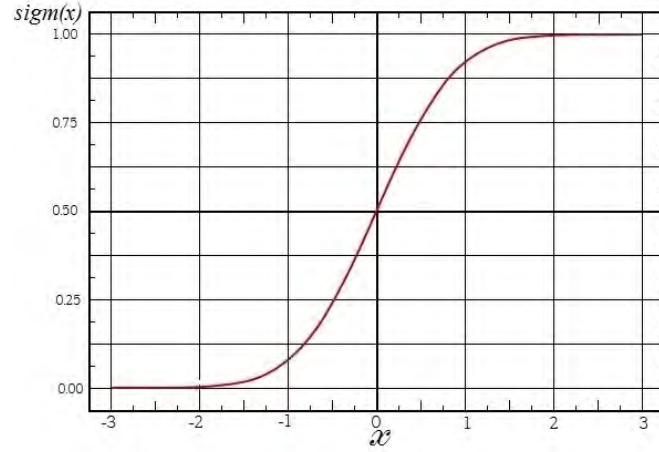


Figure 5.6. Modified Sigmoid Function

5.4.1 Functional Verification of MemTrust

The functionality of the MemTrust has been verified in a simulated environment build using GNU Octave open source software. The Sigmoid function used in the simulation has been designed using the logistic function given by Equation 5.1 [233].

$$Sigm(x) = \frac{1}{1 + e^{-x}} \quad (5.1)$$

Response time has been identified as the sample performance metric used in the simulation. The capacity of the temporary storage block for storing the recent performance values has been decided to be of thirty samples. This number has been decided based on the fact that the minimum number of samples required for maintaining normal distribution is thirty (30) [234]. Hence the thirty most recent response times have been stored on the FIFO basis by replacing the oldest sample with the most recent response time. Since the only the last thirty response time to compute the median response time, the trust score computed using this value would reflect the current status of the cloud system.

In order to maintain a consistent simulation environment, the median response time was artificially fixed during the entire duration of the simulation while the required response time was allowed to vary within a fixed interval. The trust scores computed using MemTrust was compared with that of multi-level thresholding trust computing mechanism presented in [235].

Figures 5.7 and 5.8 show the trust scores computed with MemTrust and multi-level thresholding trust computing mechanisms for constant positive and negative responses.

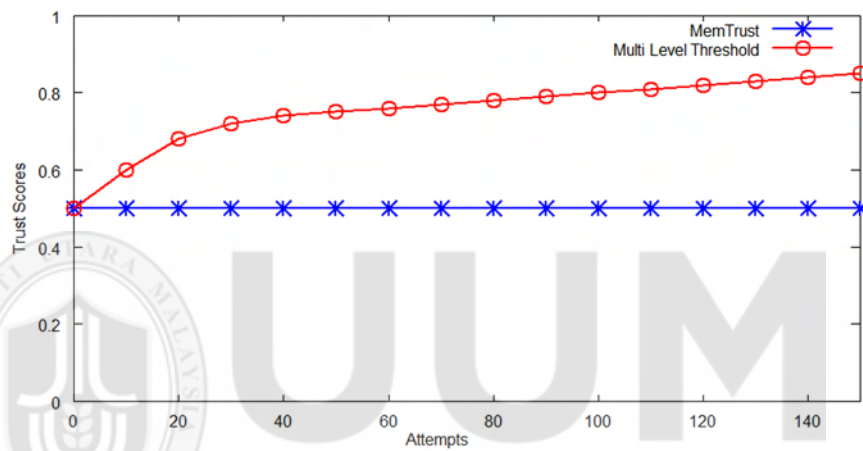


Figure 5.7. Trust Scores Computed for Constant Positive Responses

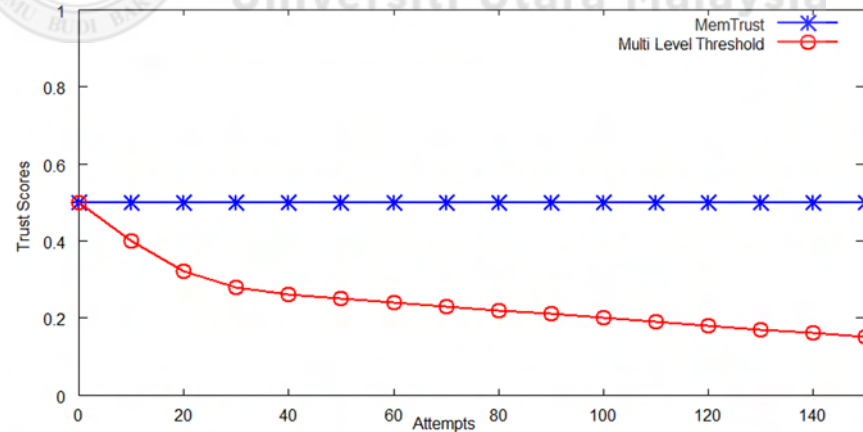


Figure 5.8. Trust Scores Computed for Constant Negative Responses

From Figures 5.7 and 5.8, it can be seen that when the constant response times were observed, the trust scores computed with the multi-level thresholding trust computing mechanism continuously adjusts the trust scores either towards 1 or 0, while the trust scores computed with MemTrust stays fixed at the same value.

Figure 5.9 shows the trust scores computed when random response time requirement was applied. Here the response times were randomly generated while maintaining the required response time fixed artificially. As the Memoryless trust computing mechanism depends only on the difference between the required response time and the real response time to compute trust scores, it shows a linear relationship with the required response time. On the other hand, the multi-level trust computing mechanism uses both the difference between the required and real response times and the previous trust scores computed for calculating the next trust score, it shows an erratic behavior. The erratic behavior is mainly due to the fact the real response times were randomly generated they occur at different times though they are shown linearly on the x-axis.

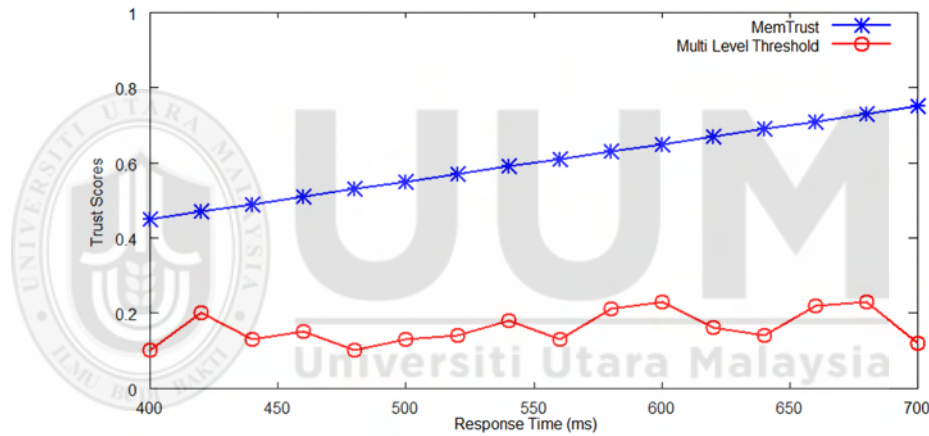


Figure 5.9. Trust Scores Computed for Random Response Time Requirement

Thus it can be observed that the MemTrust mechanism is shows stable behavior in cases of repeated requests and the trust scores depend only on the current performance of the system. Hence, it can be concluded that the MemTrust is working as per the expectations of producing a stable output in times of random erratic behavior of the system.

5.5 Hysteresis-based Trust Evolution Mechanism (HystTrust)

The HystTrust algorithm has been developed by combining the SP-SQQA and MemTrust while replacing the Sigmoid function with a hysteresis loop. Figure 5.10

shows the detailed functional diagram with individual functional units.

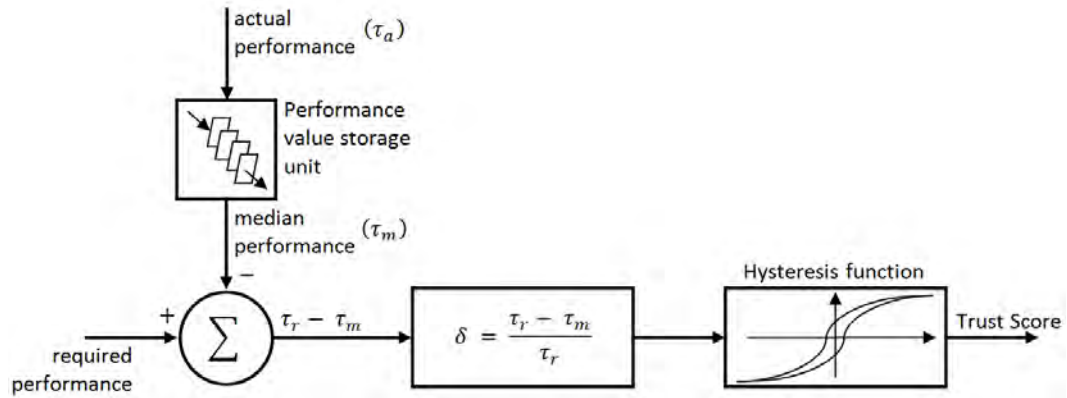


Figure 5.10. MemTrust Trust Evolution Unit

HystTrust computes the trust score using the actual performance of the system and the required performance as inputs with the aid of a hysteresis function. The new trust value is computed based on the current trust rating of the system along with the normalized service quality value (δ). HystTrust is essentially a nonlinear trust evolution algorithm. The hysteresis function has several special features that distinguish it from other nonlinear functions [236]. These features have been taken into account in many other studies including economics, electronic commerce and wireless communications [237, 238, 239]. Some of these features have been exploited in this work in order to make the trust computing mechanism more rugged and representative of the actual system performances. The special properties that are useful for trust evolution include:

- Shape of the curve (SS-shape)
- Linear operation in the middle regions
- Large input range (from $-\infty$ to $+\infty$)
- Inherent memory in the function

5.5.1 Hysteresis Function

Figure 5.11 shows the shape of a hysteresis function. From this figure, it can be seen that the output of the function not only depends on the current input but also on the history. Hence to drive the output from one extreme to the other more effort on the part of the input is required than what was required to bring it to that state originally. Hysteresis has been associated with several natural phenomena [240]. Magnetic/electrical hysteresis in ferromagnetic and ferroelectric materials, deformation of rubber or alloys under applied force, spread of epidemics in a specific geographical area and natural rate of unemployment in a given economy are few examples that show hysteresis behavior in natural systems. In order to exploit the beneficial effects of hysteresis, scientists have developed artificial systems with hysteresis behavior in order to avoid the ill effects of rapid oscillations caused by external interventions. Hysteresis based systems have been used in several engineering domains including electronics, control systems, telecommunications, mechanical engineering, software engineering etc [236]. In this research, the delayed response of hysteresis has been exploited to create a stable trust computing algorithm for cloud computing.

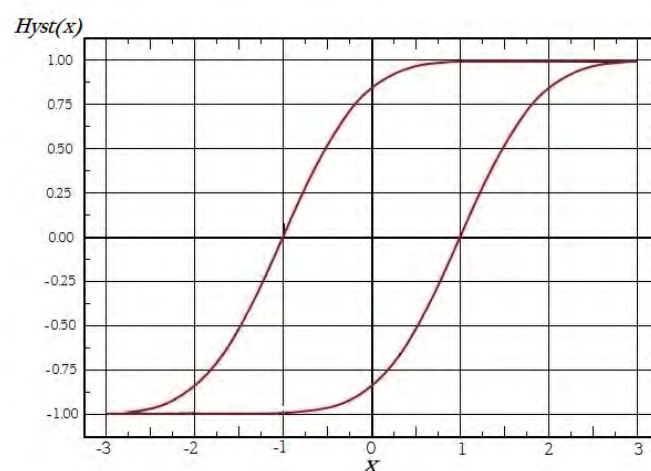


Figure 5.11. Sample Hysteresis Curve

5.5.2 Pseudo Code of the Proposed Algorithm

The pseudo code of the HystTrust algorithm is shown in under Algorithm 5.3.

Algorithm 5.3: HystTrust: Hysteresis-based Trust Evolution Algorithm

```

required response time =  $\tau_r$ 
actual response time =  $\tau_a$ 
compute the median response time =  $\tau_m$ 
compute the difference in response time  $\tau_d = (\tau_r - \tau_m)$ 
compute the normalized response time ( $\delta$ ) =  $\frac{\|\tau_d\|}{\tau_r}$ 

compute the trust score using the hysteresis function

if ( $\tau_d > 0$ ) then
     $T_{n+1} = T_n + \text{Hyst}(\delta) : T_0 = a \text{ and } n = 1, 2 \dots$ 
else
     $T_{n+1} = T_n - \text{Hyst}(\delta) : T_0 = a \text{ and } n = 1, 2 \dots$ 
end if

```

where a - initial trust score

5.5.3 Functional Verification of HystTrust

The operation of the proposed HystTrust has been functionally verified using simulations. The simulation environment was setup using Octave, the open source clone of Matlab. The hysteresis function was simulated using the Equation 5.2, which was created by combining two horizontally shifted Sigmoid functions.

$$H_{yst}(\delta_s) = \begin{cases} \frac{1}{1+e^{-(\delta_s-d)}} & \text{if } \delta \geq 0 \\ \frac{1}{1+e^{-(\delta_s+d)}} & \text{if } \delta < 0 \end{cases} \quad (5.2)$$

where δ_s - cumulative response time

d - horizontal shift

In order to simplify the simulation environment, it was decided to verify the operation using only a single performance attribute. Response time has been selected as the

attribute to be used as in the previous cases. The number of samples stored in the temporary storage was limited to thirty for computing the median response time.

During the simulation, the median response time was artificially fixed and the required response time was allowed to vary within predefined limits. Then the response times of the system were measured and the trust scores were computed using HystTrust algorithm. The trust scores thus computed are stored in a buffer as the next value to be computed would either increment or decrement the previous value based on the performance of the system. Only the last trust score needs to be stored in the buffer reducing the storage requirements of the system. It is also possible to support differentiated services with HystTrust with minor modification of the output stage. In order to support differentiated services as in the case of ACTEM, it would only be necessary to maintain multiple buffers holding trust scores for each confidence level. These trust scores corresponding to different levels of service requirements can be modified independently as explained in Chapter Four based on the performance of the system.

In order to verify the performance of the HystTrust against other algorithms, the same performance values were used to compute the trust scores using MemTrust and multi-level thresholding trust computing algorithm proposed in [235]. The results have been plotted on the same diagram. Figure 5.12 shows the trust scores computed using all four algorithms.

From Figure 5.12, it can be seen that the proposed HystTrust algorithm reduces the impact of the transient behavior of the system on the trust score. This is due to the combined effects of the statistical treatment of the input parameter through the computation of the median values and the stable behavior of the hysteresis function

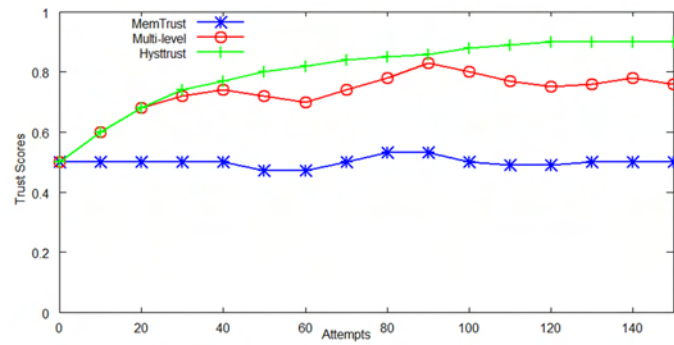


Figure 5.12. Comparison of Trust for Random Response Times

compared to the other mechanisms in the event of small fluctuations. This hysteresis based trust evolution mechanism provides a better and stable out than the other two. This proves that the HystTrust mechanism works as intended in improving the stability of the computed trust scores in the face of temporary fluctuations.

5.6 Robust Adaptive Trust Computing Mechanism (RATComM)

Robust Adaptive Trust Computing Mechanism (RATComM) has been created by combining positive aspects of the trust evolution algorithms described in Sections 5.3 and 5.5. The main components of the RATComM include the statistical validation of the inputs, trust evolution using hysteresis function and input validation controlled output enabler. The statistical validation is the process of checking if the actual performance parameter value of the service provider falls within the required confidence interval. If the actual performance (parameter value) falls within the confidence interval computed using the standard deviation and the sample mean values, the performance is considered acceptable. Otherwise, the performance of the system is taken to be below the expectations of the customer. Figure 5.13 shows the main functional units of the RATComM trust computing mechanism.

The brief operation of the RATComM is as follows: the client sends the service requests along with the committed service quality levels and the confidence intervals.

The required (committed) service quality and the corresponding confidence level are specified in the SLA signed between the service provider and the clients [241]. The actual performance of the cloud system was tracked by the trust evolution unit and the most recent performance values are stored in the temporary storage on a FIFO basis, from which the confidence interval of the system performance is computed. The observed (actual) performance is compared with the confidence interval computed to check if it falls within it or not. If the observed performance falls within the confidence interval, then the system performance is determined to be acceptable requiring no changes to the trust scores. If the observed performance is outside the confidence interval, then the trust score is modified accordingly and stored in the trust buffer using the comparator as a switch. Algorithm 5.4 lists pseudo code of the trust evolution algorithm employed in the RATComM mechanism.

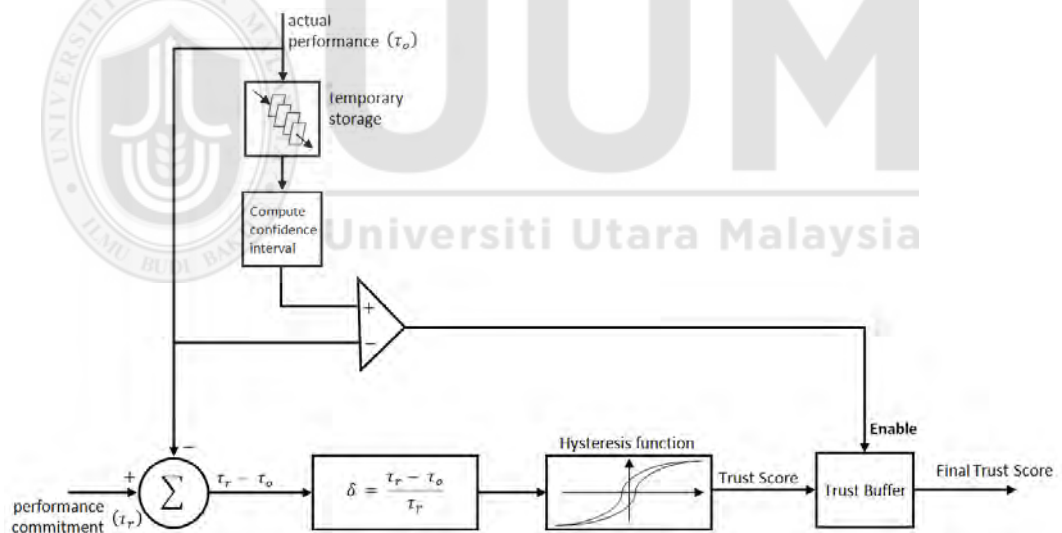


Figure 5.13. RATComM Trust Evolution Unit

5.6.1 Functional Evaluation of RATComM

The RATComM mechanism was verified for the correctness of its functionality using simulations. The simulation environment was setup using GNU Octave software. Algorithm 5.4 was implemented in Octave using response time as the performance parameter with the hysteresis function defined in Equation 5.2 to compute the final

Algorithm 5.4: RATComM Algorithm

```
Obtain required response time =  $\tau_r$ 
Obtain confidence level
Store actual response time =  $\tau_a$ 
Compute the confidence interval using the samples stored in the temporary storage
Compare the required response time with the confidence interval

if within then
    NO CHANGE
else
    Compute the difference in response time  $\tau_d = (\tau_r - \tau_a)$ 
    Compute the normalized response time  $\delta = \frac{\|\tau_d\|}{\tau_r}$ 
    Compute the trust score using the hysteresis function with  $\delta$  as input
    Store the new trust score in the output buffer
end if
```

trust score. Thirty most recent response times were stored in the FIFO buffer for computing the required statistics such as mean, median and variance values of performance, which would be used for computing the confidence interval. In order to simplify the environment and to create a set of consistent and repeatable experiments, the required response time was fixed and the actual response time was allowed to vary within limits to reflect the real world conditions.

Figures 5.14 and 5.15 show the trust scores computed using statistically validated (@ 90 and 95 percent confidence levels) inputs against that of non-validated inputs, which are the raw values that were not checked to fall within the confidence interval. From these figures, it can be observed that the trust scores computed using statistically validated inputs are more stable than that produced using non-validated inputs. In Figure 5.14, the trust scores computed using the 90% validated inputs do not change at all whereas the trust scores computed using the direct inputs fluctuate heavily. From Figure 5.14, it is very clear that even when the confidence level is relaxed, trust scores computed using validated inputs are more stable changing only when the inputs fluctuate heavily but lesser than that due to non-validated inputs.

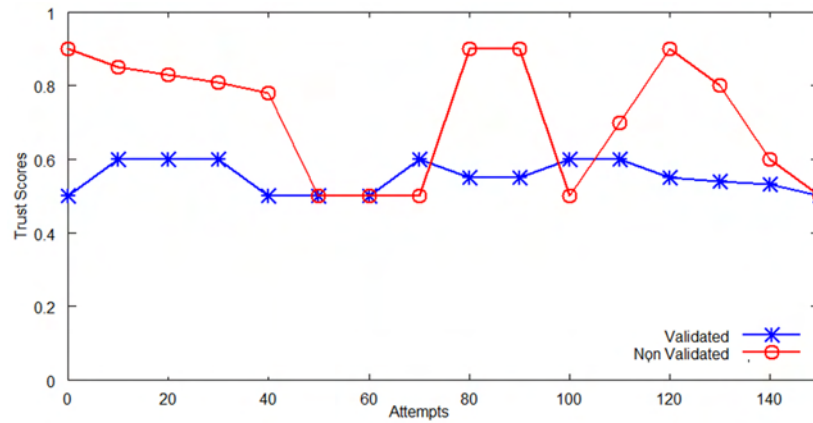


Figure 5.14. Trust Scores with 90% Validated Inputs Vs. Non Validated Inputs

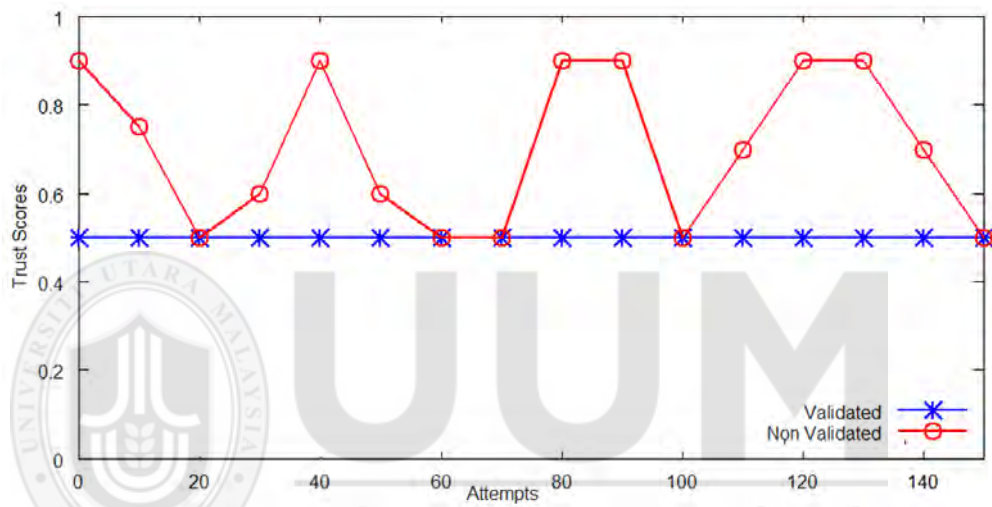


Figure 5.15. Trust Scores with 95% Validated Inputs Vs. Non Validated Inputs

Figure 5.16 shows the effect of confidence level on the trust scores computed. When the confidence level is set at 90% the fluctuation in the trust scores computed is totally eliminated compared to when the confidence level is set at 95%. Hence it can be seen that the more relaxed the requirements are more stable the trust score are even in situations of intermittent fluctuations in response time. On the other hand, when the customer demands are more stringent, the system performance must be very consistent to meet those requirements. This is clearly reflected on the trust scores computed using RATComM.

Based on the above results, it can be concluded that proposed mechanism works as

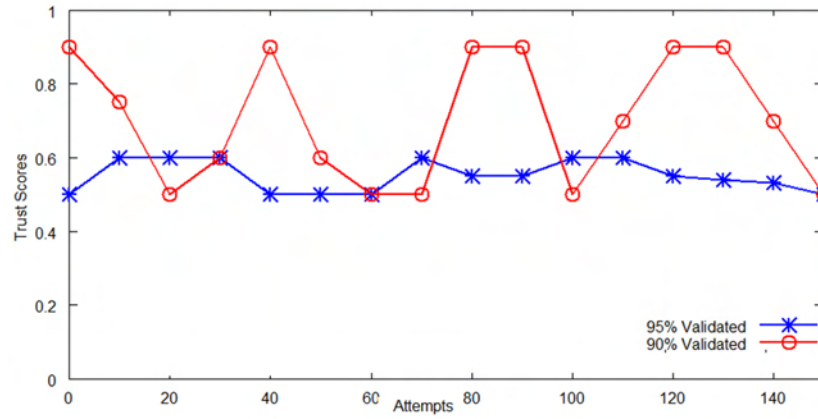


Figure 5.16. Effect of Confidence Level on Trust Scores

expected giving a stable output than that of HystTrust that does not carry out any input validation prior to computing trust scores.

5.7 Multi-Dimensional Trust Computing Mechanism (MuDTComM)

Multi-Dimensional Trust Computing Mechanism (MuDTComM) is an extension of the RATComM accepting multiple input parameters. The MuDTComM has multiple input channels for receiving and processing multiple parameters as each individual parameter needs to be differently. Figure 5.17 shows the high level design of the mechanism showing the multiple input channels, parameter conversion and combining unit and the trust computing unit in block diagrammatic format. Figure 5.18 is the detailed diagram of the same mechanism showing finer information including the buffers, statistics computing units, comparators, multiplexers, multipliers, summing points, weight handling units and trust computing module.

The operation of the mechanism is as follows: a client requests for service, it supplies the trust computing system with the required set of parameters and their required values along with the confidence level committed by the service provider. The required values are the ones committed to be met in the SLA signed between the service provider and the client. The actual performance of the system is tracked and the

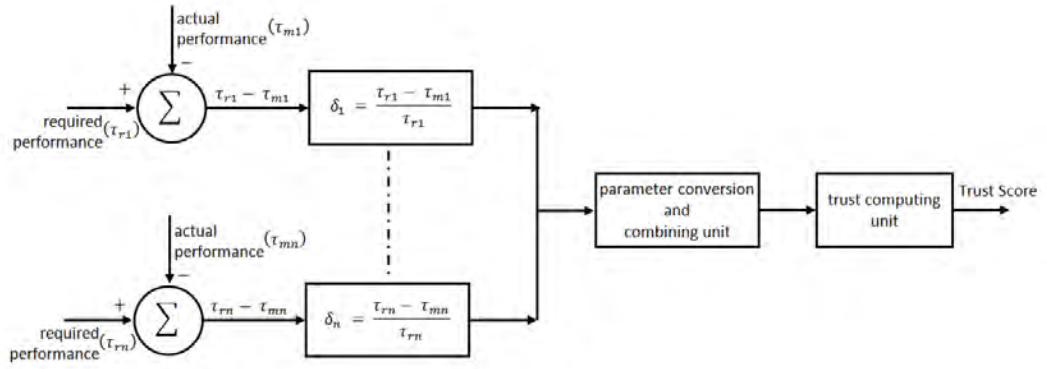


Figure 5.17. MuDTComM Trust Evolution Unit

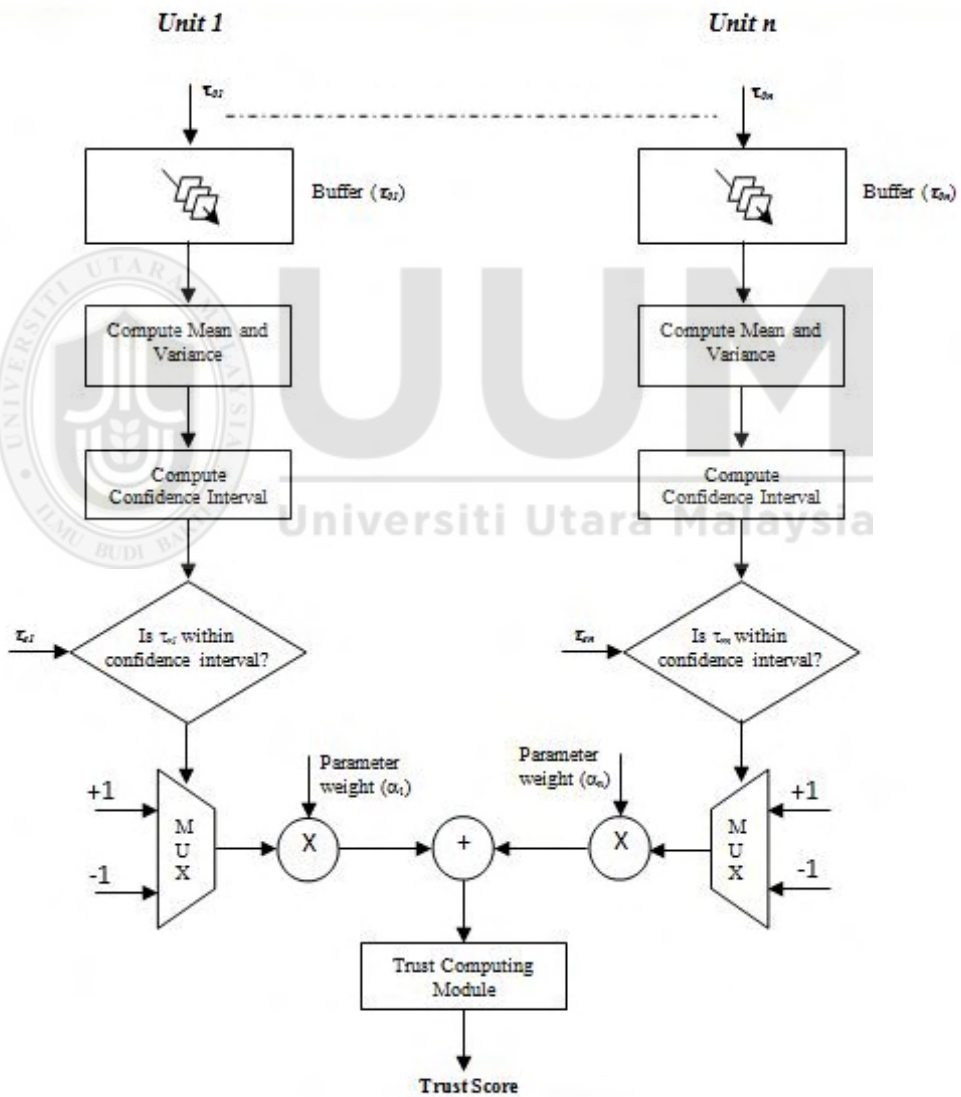


Figure 5.18. MuDTComM Trust Evolution Unit in Detail

parameter values are stored in the respective FIFO buffers.

The confidence interval each parameter using its sample mean and standard deviation will be computed using the values stored in the buffer. If the required value falls within the interval, then the performance is considered acceptable and the performance quantifier ρ is set to 1, otherwise to -1. Then the quantifiers will be weighted and summed before supplying to the trust computing unit. Finally the trust score will be computed using the hysteresis function. The pseudo code for implementing MuDTComM is given in Algorithm 5.5.

Algorithm 5.5: MuDTComM Algorithm

```

Receive confidence level
FOR (each parameter)
{
    Receive required value =  $\tau_r$ 
    Receive weight =  $w_i$ 
    Receive actual value =  $\tau_a$ 

    Store  $\tau_a$  in FIFO buffer
    Compute mean ( $\bar{\tau}_a$ ) and std. deviation ( $\sigma$ ) from sample
    Compute confidence interval for the parameter

    Check if,  $\tau_a$  with the confidence interval
    if within then
         $\rho = 1$ 
    else
         $\rho = -1$ 
    end if

    Compute the normalized performance ( $\delta$ )

    
$$\delta = \frac{\|\tau_a - \bar{\tau}_a\|}{\tau_r}$$


    Compute  $\delta_e = \rho * w_i * \delta$ 
}
Compute  $\delta_e^s = \Sigma(\delta_e)$ 
Compute Trust Score  $T = \text{Hysteresis}(\delta_e^s)$ 

```

5.7.1 Functional Evaluation of MuDTComM

Extensive simulations were carried out to verify the functionality of the MuDTComM under controlled environments. The simulation environment was setup using GNU Octave software with response time and processing time as the performance parameters. Algorithm 5.5 was implemented in Octave in such a manner that it can accept two inputs and process them. The values of input parameters, response time and processing time were restricted to create a consistent and repeatable environment. The results were compared with the trust scores computed using non-validated inputs for the stability of outputs.

Figure 5.19 shows the trust scores computed using response time as the service quality parameter by both RATComM and MuDTComM at 90% and 95% confidence levels respectively. Both mechanisms follow the same pattern at both confidence levels. This is expected as both mechanisms become same when only a single service quality attribute is considered. These mechanisms differ only in the number of parameters used for computing trust scores.

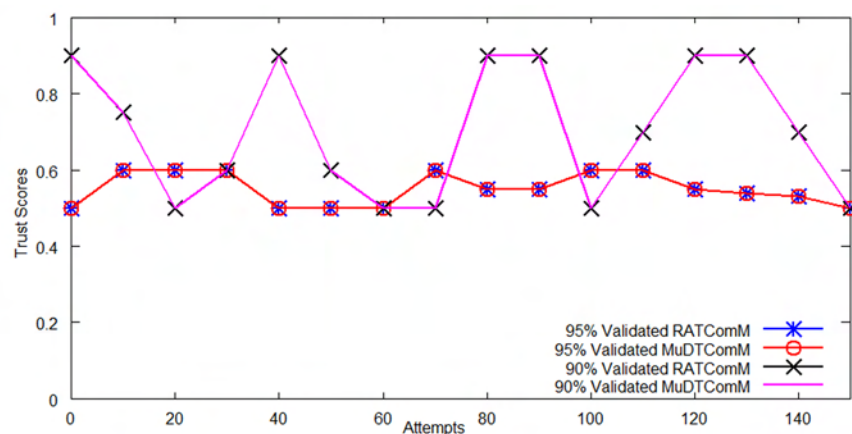


Figure 5.19. Comparison of RATComM and MuDTComM Trust Evolution Units

Figure 5.20 shows the trust scores computed using response time and service time as input parameters with equal weights at different weights at 90 percent and 95 percent confidence levels. During the 1st stage of the experiment, both response time and

service time were applied with equal weights, i.e $w_1 = w_2 = 0.5$. During the 2nd stage the weights were modified as $w_1 = 0.9$ and $w_2 = 0.1$ for response time and service time respectively.

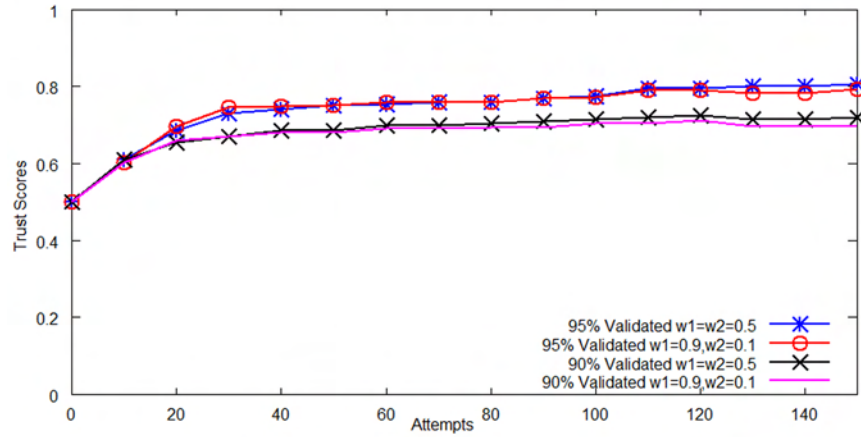


Figure 5.20. The Effect of Weights and Confidence Level on Trust Scores

From Figure 5.20, it can be observed that the confidence level plays a major role compared to the weights applied in the final trust scores computed irrespective of the number of parameters used. On the other hand, the weights play a minor role depending on the performance of the cloud system on specific parameters selected.

5.8 Summary

This chapter presented five different trust evaluation mechanisms known as ACTEM, MemTrust, HystTrust, RATComM and MuDTCOMM developed part of this research. These mechanisms were developed in an incremental fashion improving the process from one mechanism to the next on a step by stem fashion. The RATComM and MuDTCOMM were developed by combining the positive aspects of ACTEM, MemTrust and HystTrust for computing the trust scores for accommodating a single parameter and multiple parameters as inputs respectively. All the mechanisms were evaluated for functionality under limited laboratory conditions using simulations. The simulation results show that the mechanisms are functioning as expected. The performance analysis of the algorithms will be presented in Chapter Seven.

CHAPTER SIX

PROBABILITY-BASED TRUST DISTRIBUTION MECHANISM FOR CLOUD COMPUTING

6.1 Introduction

Chapter Five presented several trust evolution algorithms, which were finally combined to form two adaptive trust computing mechanisms named RATComM and MuDTComM. Though these mechanisms are sufficient for the quantification of the performance of the cloud service providers and rank them accordingly, the application of it will be limited to a small geographical area covered by single ranking unit. On the other hand, cloud services may cover large geographic areas serving a large number of customers. It is impractical to cover such a large system with a single monitoring unit. Hence it would be required to have a system of monitoring and ranking units that work together sharing the trust scores with each other. In order to facilitate the cooperatively sharing of trust scores, a trust score distributing mechanism has been developed. Chapter Six presents the reliable trust distribution mechanism that shares the trust scores between cooperating monitors. The trust scores received from other monitors are first checked for the reliability of the values based on the past experience with that particular monitor. The trust scores are then adjusted before incorporating them into the trust score table based on the past experience. This makes the trust scores to be more realistic and reliable in the presence of noise and uncertainties.

This chapter has been divided into four sections and the organization of the chapter is as follows: Section 6.1 provides a brief introduction of the chapter highlighting the importance of distribution of trust scores among the cooperating monitors. Section 6.2 presents the principles of trust distribution along with the practical issues encountered in such distribution. Section 6.3 introduces the main contribution of this chapter the

Probability-based Trust Distribution Mechanism and finally Section 6.4 summarizes the chapter.

6.2 Distribution of Trust Scores

Public cloud computing system can be considered as a large distributed system as the users and the service providers join the Internet from any location for providing as well as consuming services [242, 243]. But there exist many significant differences between traditional distributed systems and cloud computing. Under traditional definition, a distributed system is defined as a system consisting of many autonomous computers spread over a large geographical area that are working together and communicating through a network [243]. On the other hand, a cloud computing system need not have its nodes distributed over a large geographical area. The nodes of a cloud system can be hosted within a single data center but can serve a large number of customers who may be distributed over a wide geographical area [244]. Al-Roomi et al. [245] state that one of the main objectives of cloud customers is to obtain the best services in terms of service quality at a reasonable price. Hence customers may select any service provider irrespective of its physical location provided it can meet their requirements. Hence the users may be able to evaluate the service quality of any service provider irrespective of its geographical location prior selecting them. Thus it requires an independent trust management system to help customers identify the right service provider from any part of the world. It would be impractical for a single trust computing unit cover the entire Internet. A more suitable architecture would be distributed trust management system, where the individual trust computing units cover a limited area but receive and disseminate the trust computed by other units, so that the customers can get the required performance information about any cloud service provider.

The main function of the trust distribution system is to exchange the trust scores

computed by the trust computing modules described in Chapter Five. The trust distribution network proposed in this research is made of a collection of trust distribution units spread over the Internet. In principle this would make a huge peer to peer network of trust computing and distribution units.

A trust distribution unit can act as both a client as well as a server depending the requirement and situation. A node plays the role of a client when it requests the services from another node, while it would be a server, when it serves another node. These two functions can be clearly distinguished and treated as separately. Figure 6.1 shows the high-level architecture of the proposed trust distribution system.

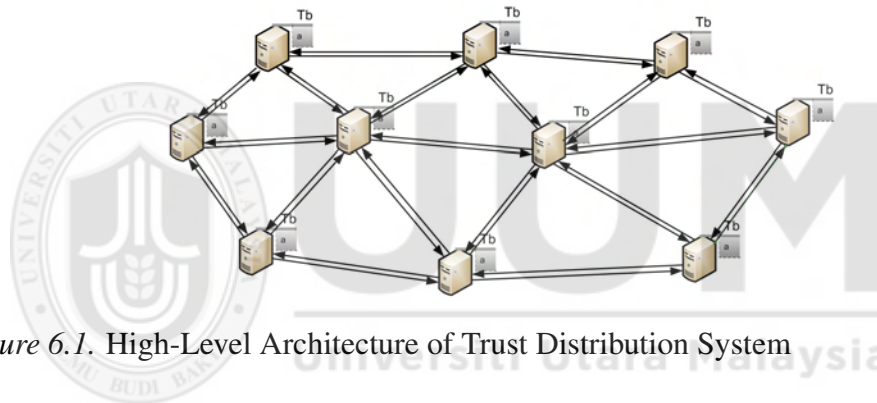


Figure 6.1. High-Level Architecture of Trust Distribution System

Figure 6.1 shows that the proposed trust distribution system has a flat architecture with only peer-to-peer nodes. This kind of flat architecture makes the system more robust and resilient against single points of failures and help system to grow and shrink independently [246]. Every node in the system maintains a trust table containing the information on the service quality of different cloud systems. These tables are populated with values either self computed by the trust computing units or received from other nodes. The single arrows running between the nodes indicate the data transfer direction and two different are used here to show that they are independent of each other. A node solicits trust scores only from its immediate neighbors and no solicitation request will be forwarded by an intermediate node to another. This strict rule is implemented in order to avoid the flooding of network with trust solicitation

messages. Figure 6.2 shows the block diagram of a trust administration unit that carries out the trust solicitation, evaluation, updating and dissemination.

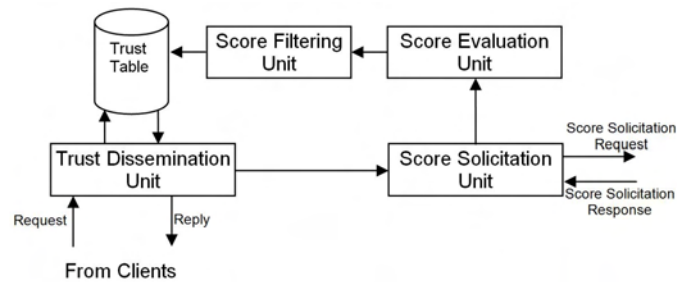


Figure 6.2. Trust Administration Unit

6.2.1 Trust Table Updating Process

The updating of the trust tables occurs under two conditions, namely the regular updates and request driven updates. The regular updates are carried out on a predetermined interval by every trust updating nodes. The request driven update is carried out when a client requests for a certain cloud resource with a given trust level. Figure 6.3 shows the trust update process in detail.

The request driven starts when a client requests for certain cloud services with specific QoS requirements. Then the trust distribution node will check its trust table for a system with the required performance characteristics (metrics). If found, the client will be informed of the details of the system. If not, the node will issue a trust solicitation notice using the flooding mechanism. When this solicitation message is received at a neighboring node, it will reply back with its trust table containing only the trust values computed by itself. The self computed trust scores are called primary scores while the ones received from a neighbor are known as secondary scores. When these scores are received at the requesting node, they will be multiplied by the trust index, which is explained later in the paper of the sending node to compute the final scores. Then the final scores are inserted into the trust table of the requester. When two or more nodes sends the trust score of a single cloud system, the score reported by the trust

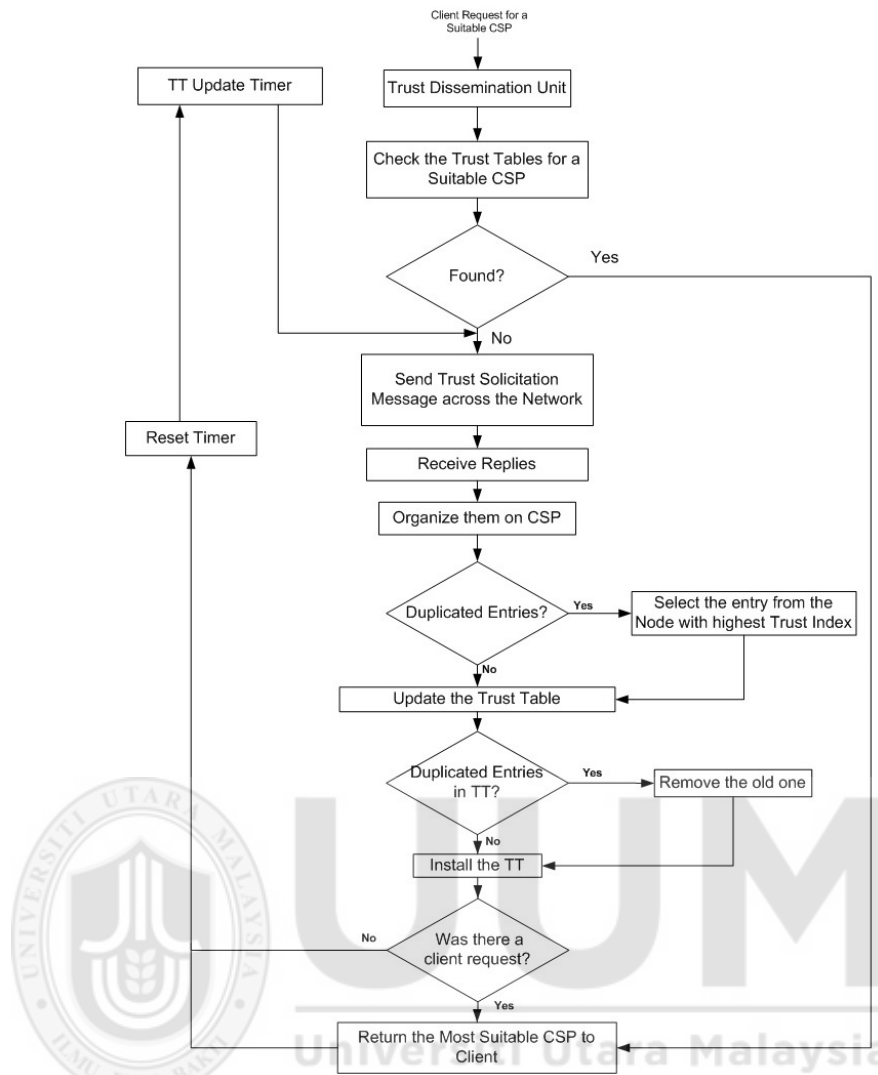


Figure 6.3. Trust Updating Process

distribution node with the highest trust index is taken for final computation.

A node sends only the primary scores as a response for score solicitation due to the following reasons:

- The reliability of the secondary trust scores is lower.
- The final trust scores computed from secondary trust scores would be very small as these scores have undergone multiple normalizations.
- If the cloud system of which the secondary information is omitted, is close to

the requesting node, there is a high chance that it will be reported as primary score by another node in the neighborhood.

- Reduces the loading of the network as only small tables are exchanged over the network.
- Limits the maximum reach of the trust scores as well as the cloud system within a specific neighborhood.

Table 6.1 shows a sample trust table. The reporting node *Self* indicates that it was computed by itself. Hence the trust index of the reporting node is taken as one (1) and the type is indicated *P* (primary). Other trust scores shown in the table are all reported by other nodes, hence the type is *S* (secondary). From Table 6.1, it can also be seen that even when all the reported trust values are equal, in this case 0.8, the values installed in the trust table varies depending on the reporting node's trust index.

Table 6.1
Sample Trust Table

Cloud service provider	Trust score	Reporting node	Reporting node's trust index	Type
CSP_1	0.80	<i>Self</i>	1	P
CSP_2	0.64	N_2	0.8	S
CSP_3	0.72	N_3	0.9	S
CSP_4	0.59	N_1	0.74	S

6.3 Probability-based Trust Distribution Mechanism (PTDiMech)

The perceived service quality received by a customer depends on the performance of the service provider as well as that of the intermediate network [247]. Hence the trust values received from the neighboring trust distribution nodes need to be adjusted for differences experienced by customers. For example, when two neighboring trust distributing nodes are located in two different parts of the Internet, they may experience differing service qualities when accessing the same services from the

same service provider due to differences in the intermediate networks, devices and systems. Some service providers may be located very close (few hops) to certain trust distribution nodes while many hops away from the other ones. Similarly the networks surrounding a trust distribution node may be more congested than the other one. Hence, when these two trust distribution nodes share the service quality attributes with each other, these external factors must be taken into account. Since, it is not practical to evaluate all the factors that may play a role in the trust scores to be shared between two nodes, it is proposed to compute a single factor known as the trust index for a node with respect the other that can combine all these difference into one.

The trust index of a node compared to another depends on its past experiences in providing performance attributes. Since a node provides the performance attributes computed by it from the data received from clients accessing these systems. The performance experienced by these clients will depend on many factors including the capability of the system, workload of the system, performance of the network and devices outside the system etc. These factors can be categorized into two groups as internal factors and external factors. The capability of a system and loading can be considered to be internal factors while the performance of the network and devices between the system and the client would constitute the external factors. Hence a system has been accessed by two clients for the same services with same performance requirements from two different parts of the Internet, the perceived performance received may differ due to the differences in the external factors. Also the perceived performance will not be a fixed one as these factors are essentially dynamic in nature. Hence the performance index of the systems needs to be computed using conditional probability based on previous experience.

A naive Bayesian network [248] can be used to represent the trust between two

trust sharing nodes based on their previous interactions. A naive Bayesian network represented by edges and vertices (nodes) is a statistical method for discovering hidden patterns in data. A naive Bayesian network consists of one root (parent) node and multiple child nodes, where the parent represents the hidden variable. The theoretical foundation of the Bayesian networks is built on the Bayes' rule for conditional probabilities [219]. Every node in a Bayesian network maintains a conditional probability table holding the relationship between the connected nodes. The naive Bayesian network built for computing the trust between two trust distribution nodes is given Figure 6.4. The parent node represents the trust (Trust Index) between the nodes while the child node represents the performance of cloud service provider whose performance parameters are reported by responding node.

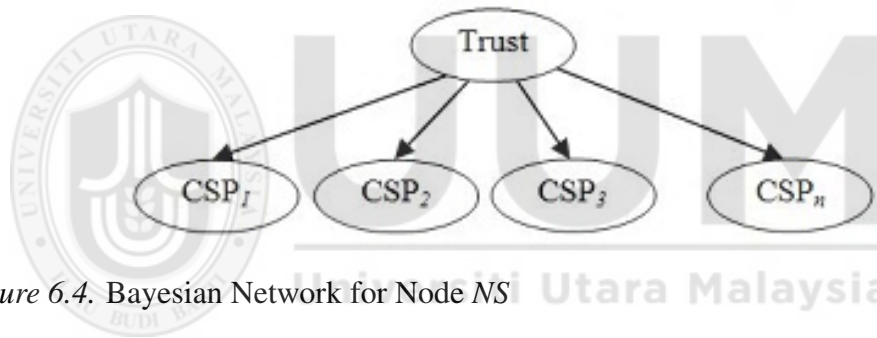


Figure 6.4. Bayesian Network for Node N_S

The root node Trust represents the confidence developed by a trust distributing node (N_S) on another trust distributing node (N_D) in providing the right information on the capabilities of cloud service providers. Each leaf node represents a cloud service provider, about whose capability the two trust distribution nodes exchange. Since a node can share the information on many cloud service providers with other nodes, the overall trust developed between these two nodes depends on the information exchanged between them and the actual performance received by a client. Based on the previous interactions, a node develops a conditional probability table for each node containing this information. Table 6.2 shows a sample conditional probability table for the node N_S developed for the node N_D from which it receives the service provider trust values.

Table 6.2
Conditional Probability Table at N_S for N_D

Cloud service provider	No of outcomes with O = 1	Probability of outcomes O=1	No of outcomes with O = 0	Probability of outcomes O=0
CSP_1	s_1	$P(CSP_1 O=1) = \frac{s_1}{S}$	n_1	$P(CSP_1 O=0) = \frac{n_1}{N}$
CSP_2	s_2	$P(CSP_2 O=1) = \frac{s_2}{S}$	n_2	$P(CSP_2 O=0) = \frac{n_2}{N}$
CSP_k	s_k	$P(CSP_k O=1) = \frac{s_k}{S}$	n_k	$P(CSP_k O=0) = \frac{n_k}{N}$

where s_i represents the positive outcomes (O=1) and n_i represents the negative outcomes (O=0) for given interaction between a cloud service provider and a client.
 S - total number of positive outcomes for all the service providers.
 N - total number of negative outcomes for all the service providers.

Table 6.2 contains the information on every cloud service provider informed by N_D to (N_S) in the form of probabilities based on its own experience. Since a trust distributing node does not interact with service providers directly, and only the clients interact with the service providers based on the recommendation of the trust provider, the performance information is obtained from the clients. When a client interacts with a service provider, it informs the trust provider about the performance, it received from the service provider. Based on this information, the performance attributes of the services provider and the node are updated according to formula given in Equation 6.1.

$$perceived\ performance = \begin{cases} \text{satisfactory,} & s_i^k = s_i^k + 1 \\ & S^k = S^k + 1 \\ \text{unsatisfactory} & n_i^k = n_i^k + 1 \\ & N^k = N^k + 1 \end{cases} \quad (6.1)$$

Overall rating (Trust Index - $TI_{D,S}$) for node N_D with respect to node N_S is computed

by combining the performance of all service providers reported by N_D . In this fashion, a nodes trust on another node is built over time.

Similarly, every node maintains a table similar to Table 6.2 for every other node that it is communicating with in the networks. Since the nodes are communicating only with the neighbors that can be reached directly, there would not be many tables to be maintained by a node.

6.3.1 Functional Evaluation of PTDiMech

The proposed trust distribution mechanism was tested using simulations. The simulation environment was setup with CloudSim, a versatile cloud computing simulation tool [197]. CloudSim provides the users with the facility to setup multiple cloud systems with many data centers equipped with variety of computers and other devices. With CloudSim, a researcher can design a simulation environment very similar to a practical system with realistic equipment and environmental conditions. Hence CloudSim has been selected as the simulation tool for testing the trust distribution mechanism PTDiMech proposed in this research. The data collected from the simulations were used to plot graphs using GNU Plot. Algorithm 6.1 shows the algorithm used to verify the functionality of PTDiMech. The functionality was verified only for one parameter, namely the response time. Though it was tested for one parameter, the mechanism can accommodate any number of parameters with minor modifications to the algorithm. The evaluation of the algorithm for its validity will be carried out using more than one parameter in Chapter Seven.

Algorithm 6.1 was tested extensively for functionality under different conditions. Figure 6.5 shows the change in trust scores over a period of time for a given cloud service provider. The initial trust value was set at 0.8.

Algorithm 6.1: Probability-based Trust Distribution Algorithm

required response time = τ_r
required confidence level = C_r
receive the trust score (for the above given response time) = T_r
actual response time = τ_a
Compute confidence interval - C_I
if (τ_a within C_I) **then**
 $S = S + 1$: (increment S)
else
 $F = F + 1$: (increment F)
end if

$$\text{Probability of Success } P_s = \frac{S}{(S+F)}$$

Compute the new trust value T_n
 $T_n = P_s * T_{n-1} : T_0 = T_r$ and $n = 1, 2, \dots$
where, S – No. of successes and F - No. of failures

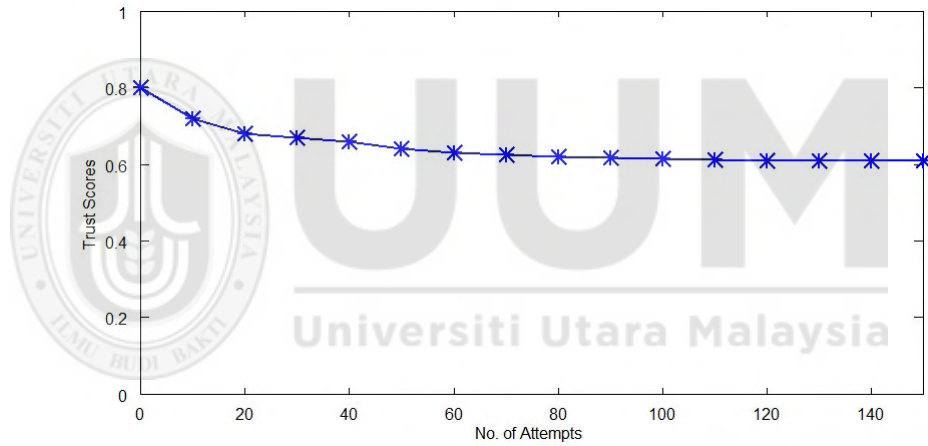


Figure 6.5. Change of Trust Scores for CSP_1 over a Period of Time

From Figure 6.5, it can be seen that though the result scores show some fluctuations at the beginning, it settles down at a fixed value in the long run. This is due to the stable performance of the remote cloud system as the fraction of positive responses (meeting the commitment for a given response time) stabilizes in the long run.

6.4 Summary

This chapter presented the probability-based trust distribution mechanism for cloud computing developed as part of this research. The proposed mechanism was built

from basic principles using Bayes' theorem. The functionality of the mechanism was tested under limited laboratory conditions using simulations. The simulation results show that the mechanism functions as expected. The performance analysis of the mechanism will be presented in Chapter Seven.



CHAPTER SEVEN

PERFORMANCE ANALYSIS OF TRUST COMPUTING AND DISTRIBUTION MECHANISMS

7.1 Introduction

Chapters Four, Five and Six presented the trust quantification, trust evolution and trust distribution mechanisms that are the main contributions of this research. Combining these three different mechanisms, it is possible to build a service quality quantification system that can monitor and categorize (rank) cloud service providers according to their performance. Chapter Seven presents the performance analysis of these mechanisms along with the details of the experiments carried out. The performance of each mechanism proposed in this research is compared against that of the mechanisms proposed in the literature under similar conditions.

The organization of Chapter Seven is as follows: Section 7.1 provides a brief introduction to the chapter along with the organization. Section 7.2 presents the simulation environment in detail along a justification for using such an environment. Sections 7.3, 7.4 and 7.5 explain the performance analysis of service quality quantification, trust computing and trust distribution mechanisms respectively. Finally Section 7.6 summarizes the chapter highlighting the important points discussed in herein.

7.2 Simulation Environment

The main aim of the performance analysis stage is to build an environment similar to that is encountered in the real world on the selected simulator and run the predetermined scenarios. The results obtained from these experiments can be analyzed to check the performance of the mechanism or model that has been under investigation.

This research is concerned about measuring and quantification of the service quality of cloud service providers from the customers' perspective and not to modify the performance of the cloud services in any way. Hence the cloud systems used in simulations were created similar to that are available in the market.

The simulator along with the required other software were downloaded and installed on a computer with the specification and configuration given in Table 7.1.

Table 7.1
Specification of the Host Computer

Component	Specification
Processor type and class	Intel Core i3 M350 @ 2.27GHz
No. of cores	4 with hyper threading
Memory (RAM)	8 GB
Operating System	Windows 7 Professional
Java version	Standard Edition, Version 8 Update 25
IDE	Eclipse IDE for Java Developers
IDE Version	Mars Release (4.5.0)

The Eclipse Integrated Development Environment (IDE) was used as CloudSim lacks the Graphical User Interface (GUI). Figure 7.1 shows the Eclipse IDE loaded with CloudSim simulation environment.

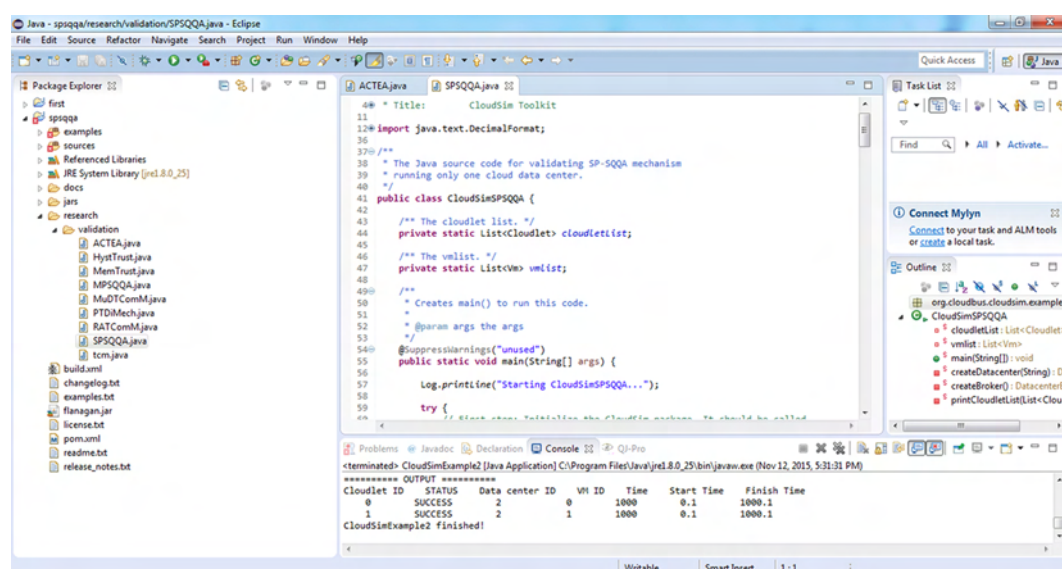


Figure 7.1. Eclipse IDE Loaded with CloudSim

Once the CloudSim environment has been setup and tested, the individual configurations suitable for each and every scenario need to be run. CloudSim simulation goes through a specific set of specific stages known as the CloudSim life cycle [249]. Figure 7.2 shows a typical CloudSim life cycle.

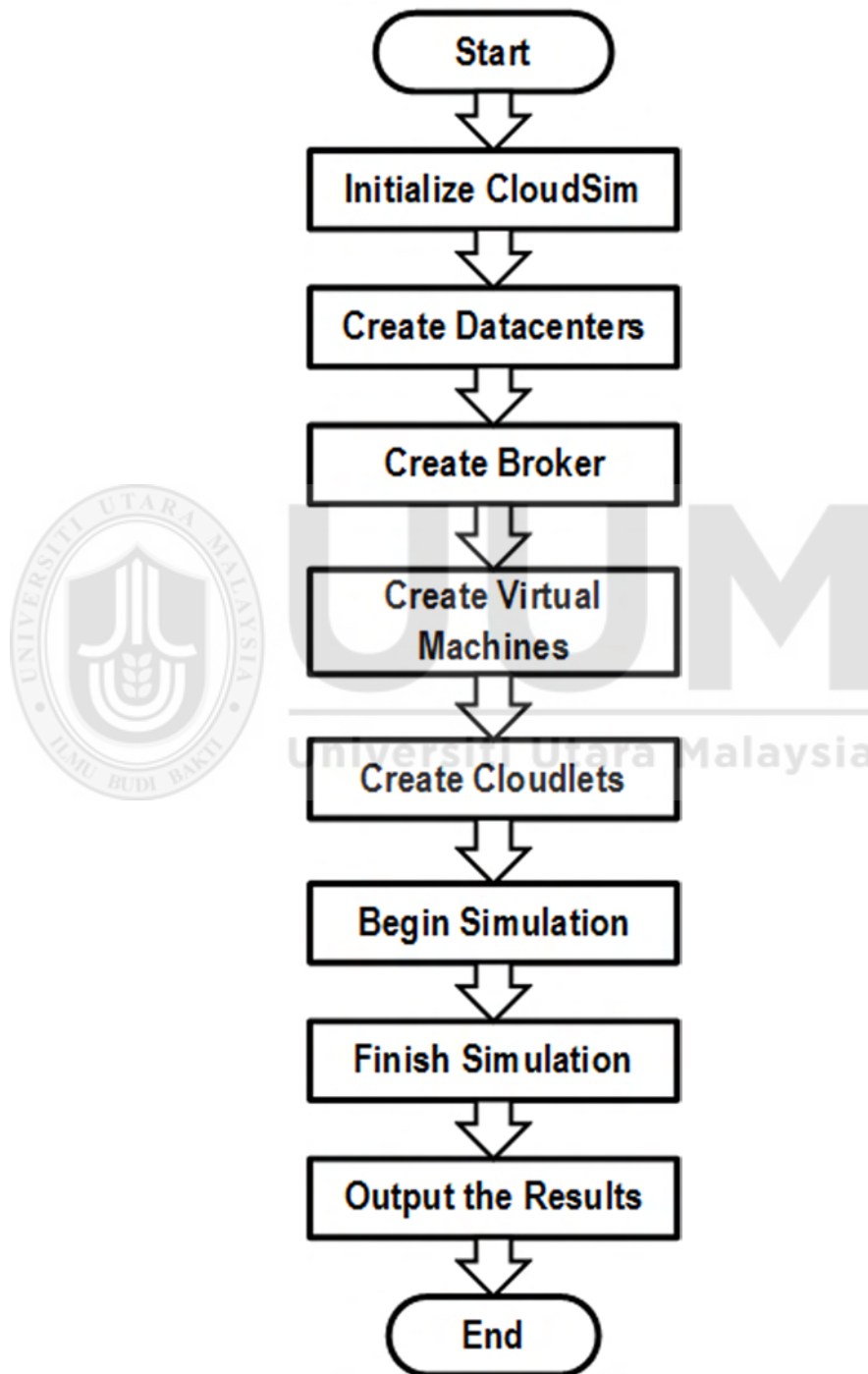


Figure 7.2. A Typical CloudSim Life Cycle

7.3 Performance Analysis of Service Quality Quantification Mechanisms

Under Service Quality Quantification Mechanisms, both SP-SQQA and MP-SQQA were tested. For these experiments, a cloud data center was setup is given in Table 7.2.

Table 7.2

Experiment Setup Attributes for the Evaluation of Service Quality Quantification Mechanisms

Attribute	Sub-Attribute	Value
No. of data centers		1
No. of servers		5
Server properties	Architecture	x86
	No. of processors	1 (Quad Core)
	Memory (RAM)	8 GB
	Storage	1 TB
	Network bandwidth	1 Gbps
VM allocation policy		Simple
VM scheduling method		Time and space shared
Cost		0

The service quality requirements of the clients were then set based on the predetermined parameters of response time, service time and availability and other resources. Table 7.3 shows the service quality requirements set in for these experiments.

Table 7.3

Service Quality Requirements for Service Quality Quantification Mechanisms

Parameter	Sub-Parameter	Value
Response time		500 ms
Service time		10 minutes
Availability		100%
Computing Power	Processor	2.27 GHz Clock speed 6 MB Cache 533 MHz Bus speed
	Memory	2 GB
	Storage	100 GB
Network speed	Bandwidth	100 Mbps
	Latency	3.33 ms

7.3.1 Performance Analysis of SP-SQQM

The SP-SQQM has been tested and validated through a set of extensive experiments carried out in a simulated environment. Validation of the results has been carried out by comparing the results with that of Combined Trust Model and QoS Trust Model proposed in [48] and [49] respectively.

Combined Trust Model proposed in [48] has been designed by combining three different trust models such as Identity-based trust (T_I), Capability-based trust (T_C) and Behavior-based trust (T_B). They have been combined as shown in Equation 7.1.

$$CT = a * T_I + b * T_C + c * T_B \quad (7.1)$$

where a, b and c are fractional weights with $a + b + c = 1$.

QoS Trust Model proposed in [49] is a method for combining different performance parameter into a single quantity. The QoS Trust has been defined as shown in Equation 7.2.

$$QT = w_1 * P_1 + w_2 * P_2 + w_3 * P_3 + w_4 * P_4 \quad (7.2)$$

where w_1, w_2, w_3 and w_4 are fractional weights with $w_1 + w_2 + w_3 + w_4 = 1$.

and

P_1, P_2, P_3 and P_4 are different performance parameters.

Prior to collecting the data for analysis, the simulation was allowed to run freely for about 10 minutes in order to let it become stable. By allowing the system to stabilize,

it would be possible to avoid temporary fluctuations and misleading results.

The experiment consisted of 10 job submissions starting with 1000 jobs for the first submission and increasing the number of jobs per consecutive submission by another 1000 jobs. Thus the last job submission makes the total number of job submissions to be 10,000 altogether cumulatively. Each job submission should create a single virtual machine for every 1000 jobs and submit these jobs to that virtual machine. The processing of the jobs is simulated by creating a cloudlet and running it for a predetermined time of 10 minutes as stated above. Figures 7.3, 7.4 and 7.5 show the trust scores computing using SP-SQQA, QoS Trust Model and Combined Trust Model for response time, service time and availability respectively. Since a single parameter was considered for computing the trust scores, the weights for the given specific parameters were set to 1 and 0 for other two parameters in the QoS Trust Model and Combined Trust Model in this experiment.

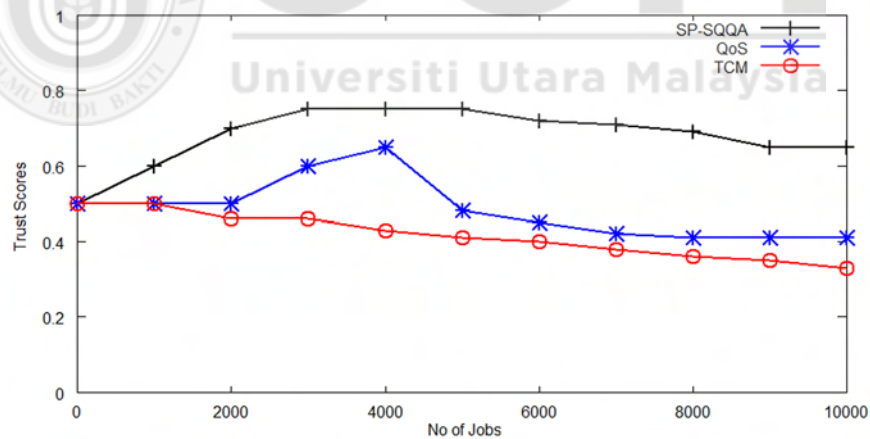


Figure 7.3. Trust Scores Computed Using Response Time

From Figures 7.3, 7.4 and 7.5, it can be seen that the trust scores computed using the proposed SP-SQQA has stable performance compared to other two mechanisms for all three performance parameters selected. One of the most important aspects to note from the figures, is the absence of erratic behavior in the trust scores computed by the proposed SP-SQQA. The trust scores improves initially and then either settles

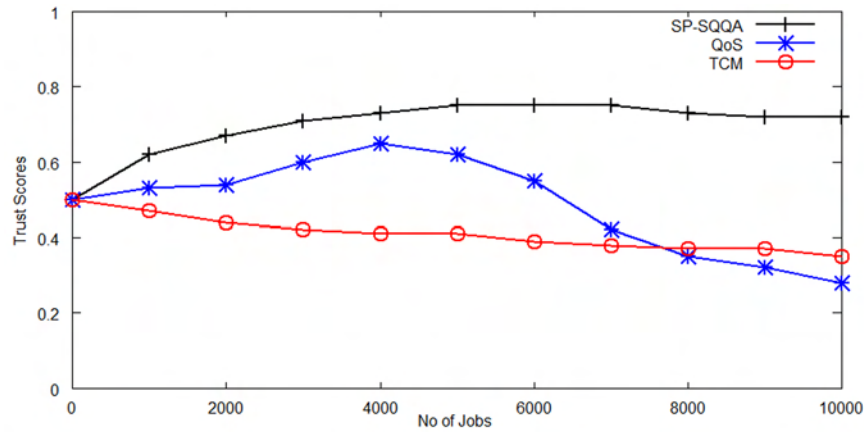


Figure 7.4. Trust Scores Computed Using Service Time

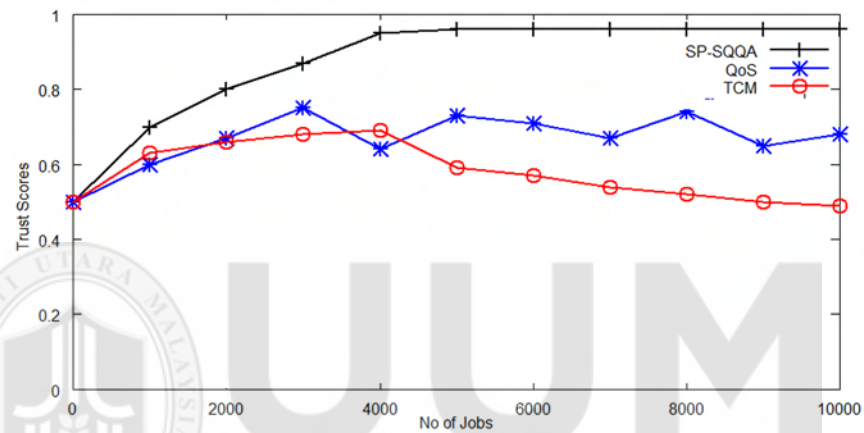


Figure 7.5. Trust Scores Computed Using Availability

down or shows slowly dropping as the loading is increased. This is the expected behavior as when the cloud system has sufficient resources to service the requests, they can either meet or exceed the customers' requirements. This should improve the trust rating of the providers. On the other hand, both TCM and QoS models show very erratic behavior throughout. The QoS trust model fluctuates around a trust score without settling on that value due to instantaneous performance fluctuations and TCM model continue to deteriorate without showing any stable performance. The stability of the SP-SQQA results from the fact that the temporary service provider performance fluctuations are suppressed through statistical validation process.

7.3.2 Performance Analysis of MP-SQQM

The performance of the MP-SQQM was also verified through extensive simulations carried out using CloudSim. The experimental setup was similar to that of the SP-SQQM but the multiple parameters were taken together for computing the trust scores. The simulation results were validated by comparing them against the performance of the QoS trust computing model and Combined Trust Model proposed by in [49] and [48] respectively. Figure 7.6 shows the trust scores computed with equal weights applied to all three selected service quality parameters, namely response time, service time and availability.

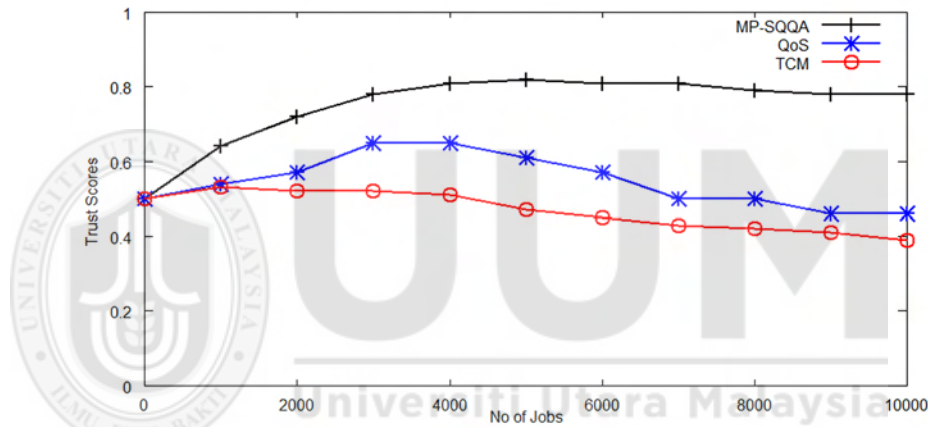


Figure 7.6. Trust Scores with Equal Weights

From Figure 7.6, it can be seen that the trust scores computed using the proposed MP-SQQM initially improves and settles down at the specific value. On the other hand, the trust scores computed using other two mechanisms show erratic behavior at the beginning and tend to settle down way below that of MP-SQQM.

Figures 7.7 and 7.8 show the trust scores computed using all three mechanisms using different weights for the service quality parameters. Figure 7.7 is the result, when the weights of 0.5, 0.25 and 0.25 were applied to response time, service time and availability respectively. On the other hand, Figure 7.8 shows the outcome with weights 0.2, 0.3 and 0.5 applied to response time, service time and availability

respectively.

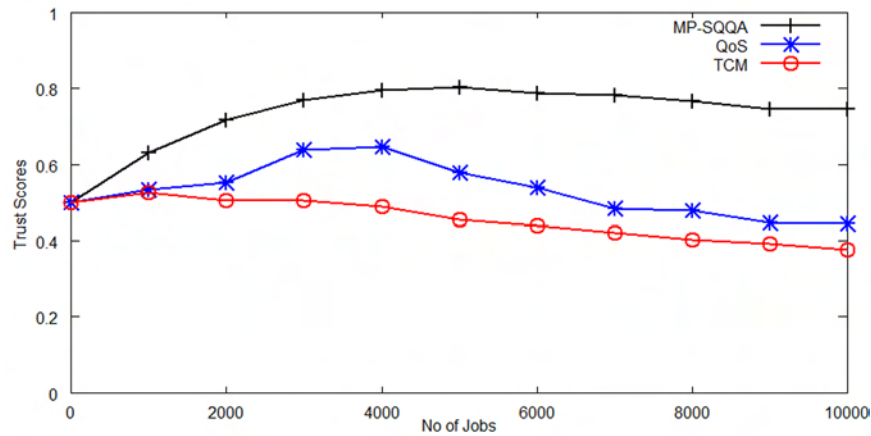


Figure 7.7. Trust Scores with Unequal Weights (Case I)

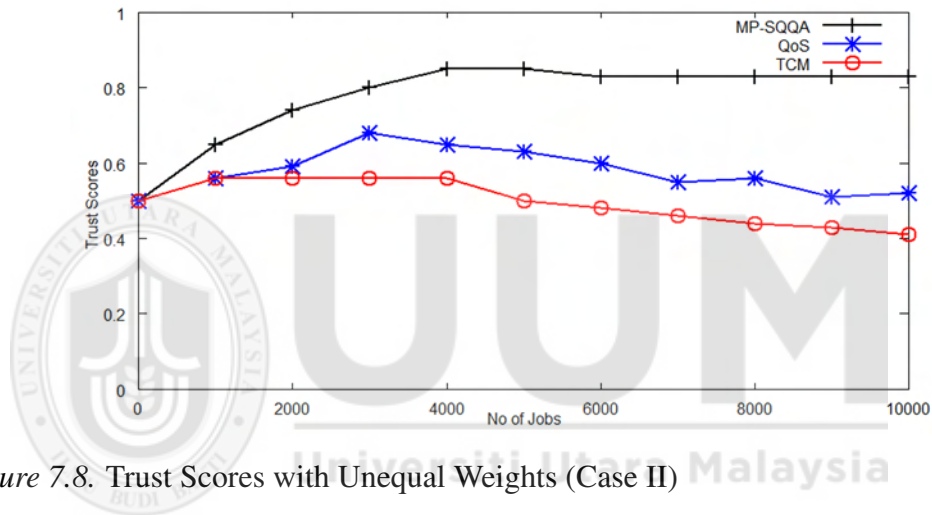


Figure 7.8. Trust Scores with Unequal Weights (Case II)

From Figures 7.7 and 7.8, it could be observed that the trust scores computed using the mechanism proposed in this research, that is MP-SQQA performs shows faster and more stable performance than both QoS trust computing mechanism and Combined Trust Computing mechanisms. Reaching the final trust score faster enables the customers to identify the real capability of the systems in a short time. While the stable trust scores create confidence in customers on the performance of the systems as a one that can be trusted to give stable and consistent performance.

Most importantly, it must be noted that the proposed trust computing mechanism improves the trust scores gradually and smoothly when the performance of the cloud system is better than or equal to the expectations of the customer as specified in

the SLA. But the behavior of both QoS trust computing mechanism and that of the combined trust computing mechanism are erratic in changing the trust scores. The QoS trust computing mechanism increases the trust scores in an ad hoc manner and then drops very fast to a value below the middle value of 0.5 giving the indication that the system performance is unacceptable. The performance of the combined trust computing mechanism is worse as it never improves the trust scores but continuously diminished it from the beginning. This is against the expectation that the computer system has acceptable performance at least at the beginning, with low loading.

Hence it can be concluded that the proposed trust computing mechanism performs better than both QoS trust computing mechanism and combined trust computing mechanism. This validates the performance of the MP-SQQM as the better mechanism of all.

7.4 Performance Analysis of Trust Computing Mechanisms

Similar to performance quantification mechanisms, trust computing mechanisms were also tested using simulations. The simulation environment was setup similar to that of trust quantification mechanisms explained in Section 7.3 with one data center containing only one cloud server. Only one server was installed as performances of these mechanisms are required to be evaluated under different loading conditions. Thus the configuration of the data center is given in Table 7.4.

The service quality requirements of the clients were then set based on the performance parameters of response time, service time and availability and other resource requirements. Table 7.5 shows the client requirements set in for these experiments.

Table 7.4

Experiment Setup Attributes for the Evaluation of Trust Computing Mechanisms

Attribute	Sub-Attribute	Value
No. of servers		1
Server properties	Architecture	x86
	No. of processors	1 (Quad Core)
	Memory (RAM)	8 GB
	Storage	1 TB
	Network bandwidth	1 Gbps
VM allocation policy		Simple
VM scheduling method		Time and space shared
Cost		0

Table 7.5

Service Quality Requirements for Trust Computing Mechanisms

Parameter	Sub-Parameter	Value
Response time		500 ms
Service time		10 minutes
Availability		100%
Computing Power	Processor	2.27 GHz Clock speed
		6 MB Cache
		533 MHz Bus speed
	Memory	2 GB
	Storage	100 GB
Network speed	Bandwidth	100 Mbps
	Latency	3.33 ms

Several rounds of experiments were carried out under different loading conditions to test and validate the performance of the trust computing mechanisms proposed in this research. The results were validated by comparing them with that of the fuzzy theory based trust computing mechanism proposed by Gu et al. in [50]. The fuzzy theory based trust computing mechanism inquires other users and other cloud service providers for information on the trustworthiness of the selected service provider. Hence this is basically an opinion based trust computing mechanism. In the experiments carried out in this research, fuzzy theory based mechanism has been slightly modified. The experiments carried out in this research takes the past

performance values of the service providers in place of the opinions of others. This way it is possible to avoid the misrepresentation of facts by rogue users and service providers as well as the independence of the trust computing monitor nodes are maintained. The other shortcoming of the model proposed by Gu et al. is the inability to distinguish the service quality factors from the opinions. By considering the past data stored in the monitor nodes, it is also possible to compute the trust scores based on specific parameters.

Similar to the experiments carried out for testing the service quality quantification mechanisms, these experiments also consisted of 10 job submissions starting with 1000 jobs for the first submission and increasing the number of jobs per consecutive submission by another 1000 jobs. Thus the last job submission contained 10,000 jobs altogether cumulatively. Each job submission should create a single virtual machine for every 1000 jobs and submit these jobs to that virtual machine. The processing of the jobs is simulated by creating a cloudlet and running it for a predetermined time of 10 minutes as stated above.

7.4.1 Performance Analysis of Adaptive Continuous Trust Evolution Mechanism

The performance of the Adaptive Continuous Trust Evolution Mechanism (ACTEM) was tested under different loading conditions as above. In order to obtain the right performance values the cloud system under test was loaded to the predefined loading levels and then the required service request was sent. Only the performance values for the above specific service request were measured and plotted. As explained in Chapter Three, only response time, service time and availability were measured in these experiments.

Figure 7.9 shows the trust scores computed using response time as input. The initial trust was assumed to be 0.5 to indicate a neutral value. ACTEM was used to compute trust scores at 90, 95 and 99 percent confidence levels. The trust scores computed using fuzzy theory is also plotted in the same graph for the comparison purposes.

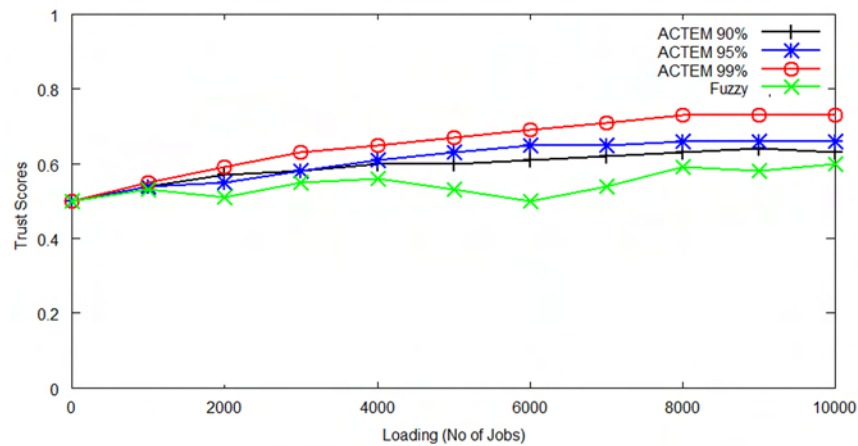


Figure 7.9. Trust Scores Computed based on Response Time

From Figure 7.9, it can be seen that the trust scores computed using the proposed mechanism show smooth improvement while fuzzy theory based trust scores shows fluctuations. This is mainly due to validation and non-validation of inputs by the respective methods. The confidence level also plays role in the final trust scores computed as the interval for validation depends on it. Overall the proposed mechanism performs better than the fuzzy theory based mechanism.

Figure 7.10 shows the trust scores computed using service time as input parameter. Similar to the previous case, ACTEM was used to compute trust scores at 90, 95 and 99 percent confidence levels. The same plot shows the trust scores computed using fuzzy theory too. From Figure 7.10, it can be seen that ACTEM performs better than fuzzy theory based trust computing mechanism even for response time as input parameter. Figure 7.11 shows the trust scores computed using availability as input parameter. The same graph shows the plots of ACTEM computed trust scores at 90, 95 and 99 percent confidence levels and that of fuzzy theory.

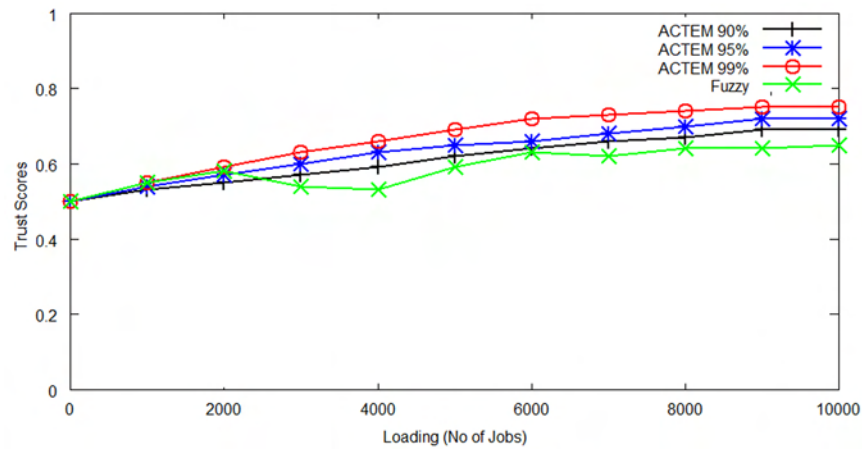


Figure 7.10. Trust Scores Computed based on Service Time

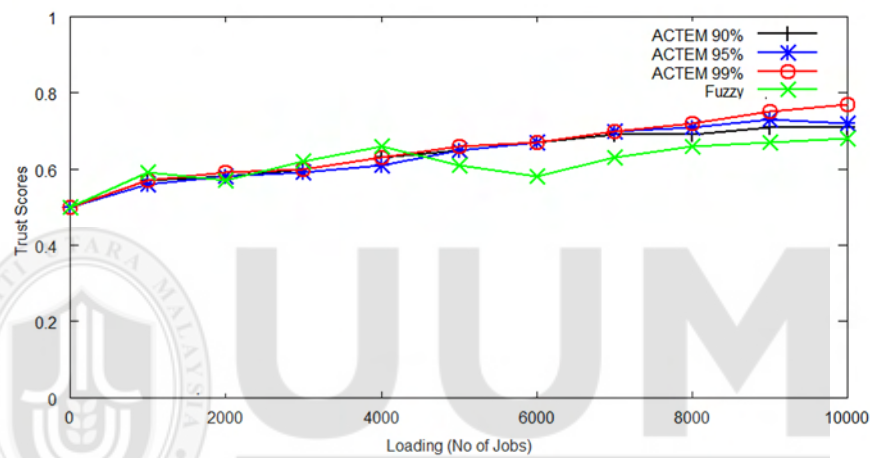


Figure 7.11. Trust Scores Computed based on Availability

From Figure 7.11 show that the fuzzy theory based trust scores a little better performance at the low loading levels. But this trend is not maintained when the loading was increased. On the other hand the trust scores computed using ACTEM show smooth performance throughout.

From the above observations, it can be concluded that the Adaptive Trust Computing Mechanism performs better than the fuzzy theory based method. Also the ACTEM provides the users the ability to state their requirements based on confidence level too. This is more practical than computing a single score for all the users irrespective of the stringency of the needs.

7.4.2 Performance Analysis of MemTrust

The Memoryless Trust Computing (MemTrust) was also tested under the same conditions as ACTEM described in Sub Section 7.4.1. Figures 7.12, 7.13 and 7.14 show the trust scores computed using the proposed MemTrust and fuzzy theory based mechanisms based on response time, service time and availability respectively.

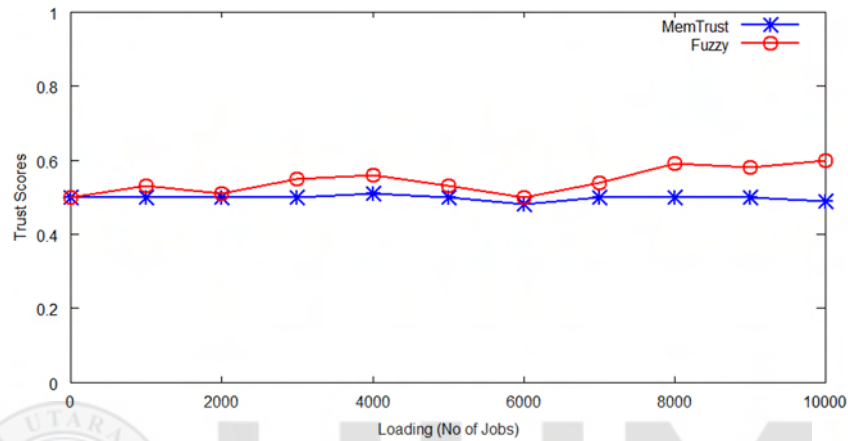


Figure 7.12. Trust Scores Computed based on Response Time

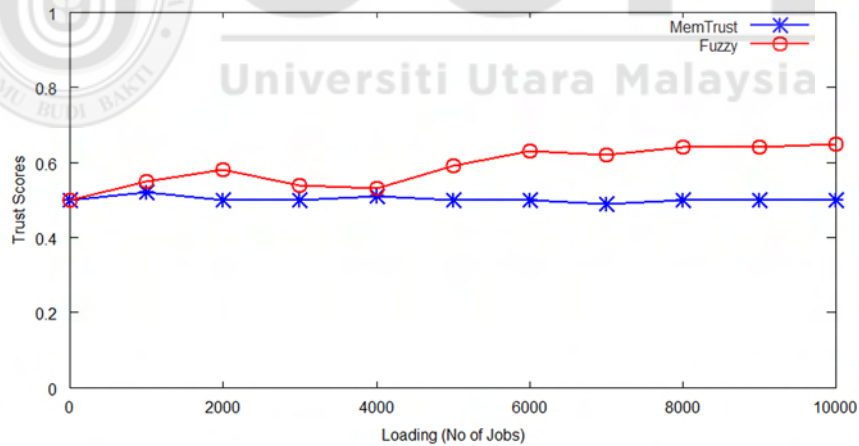


Figure 7.13. Trust Scores Computed based on Service Time

From Figures 7.12, 7.13 and 7.14, it can be seen that the trust scores computed using the proposed MemTrust mechanism always tend to stay close to the middle value of 0.5 while the fuzzy theory based trust computing mechanism updates the trust scores. This is due to the reason that the MemTrust mechanism uses only the current value to compute the trust scores. It does not take the previous performance into account

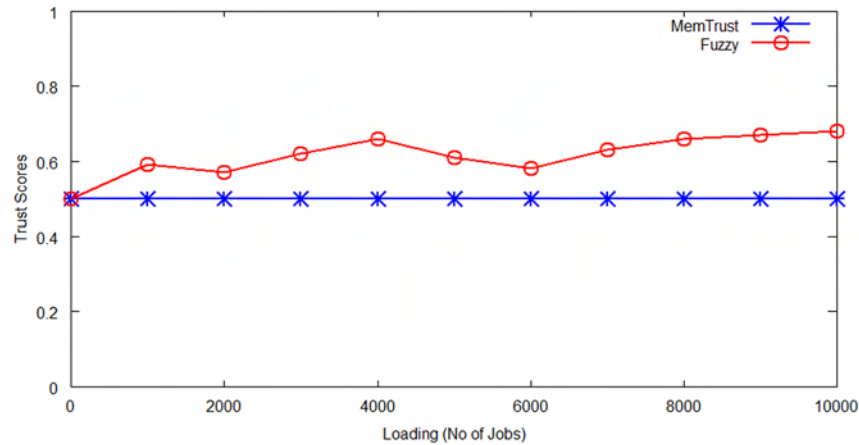


Figure 7.14. Trust Scores Computed based on Availability

to avoid itself from misrepresentation of performance information by adversaries or malicious users. As the trust scores depends only on the current performance of the cloud computing node, the trust scores cannot be forced towards extreme values by repeated misrepresentation of performance information. However there is no reward for well performing systems either. Though, the trust scores computed using this mechanism does not represent the true behavior of the cloud systems, this mechanism acts as the basis for the strong and stable trust computing mechanisms developed in this research.

7.4.3 Performance Analysis of HystTrust

The same environment has been used to evaluate the Hysteresis-based Trust Evolution Mechanism (HystTrust). Figures 7.15, 7.16 and 7.17 present the trust scores calculated with the proposed HystTrust and fuzzy theory based mechanisms based on response time, service time and availability respectively.

From the above figures, it can be seen that the proposed hysteresis-based trust computing mechanism performs better than the fuzzy theory based trust computing mechanism throughout. Also, it can be observed that the output of the HystTrust mechanism is more stable in the face of small temporary fluctuations. This is mainly

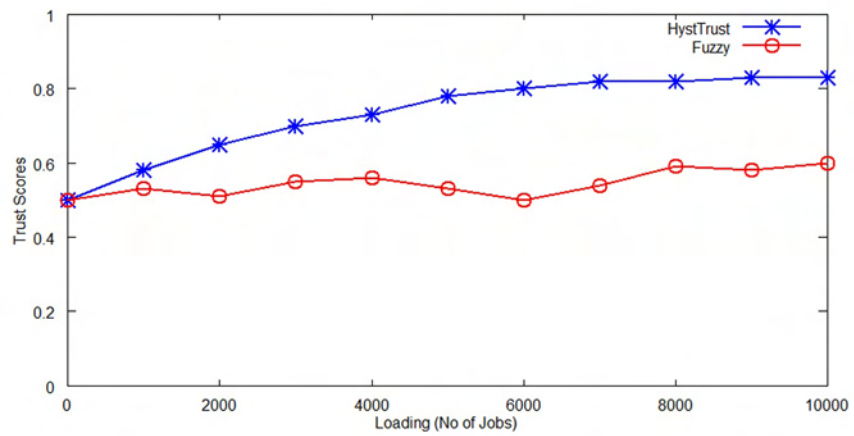


Figure 7.15. Trust Scores Computed based on Response Time

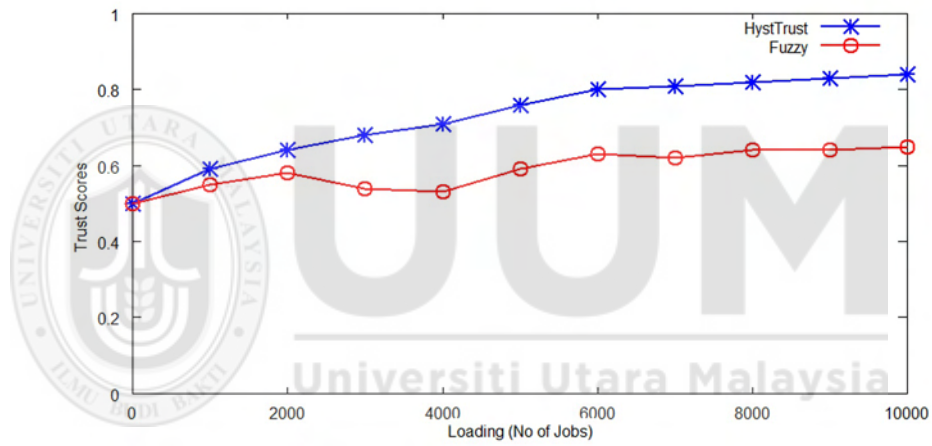


Figure 7.16. Trust Scores Computed based on Service Time

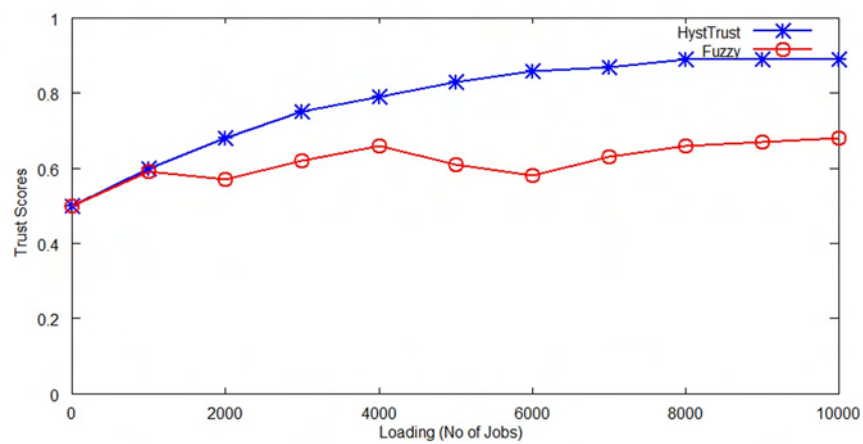


Figure 7.17. Trust Scores Computed based on Availability

due to the inherent memory included within the hysteresis functions that avoids the fluctuations of output for small changes in the input. Thus, it can be concluded that the proposed hysteresis-based trust computing mechanism has better performance than the fuzzy theory based trust computing mechanism and provides more stable output.

7.4.4 Performance Analysis of RATComM

The same environment has been used to evaluate the Robust Adaptive Trust Computing Mechanism (RATComM). Figures 7.18, 7.19 and 7.20 present the trust scores calculated with the proposed RATComM and fuzzy theory based mechanisms based on response time, service time and availability respectively.

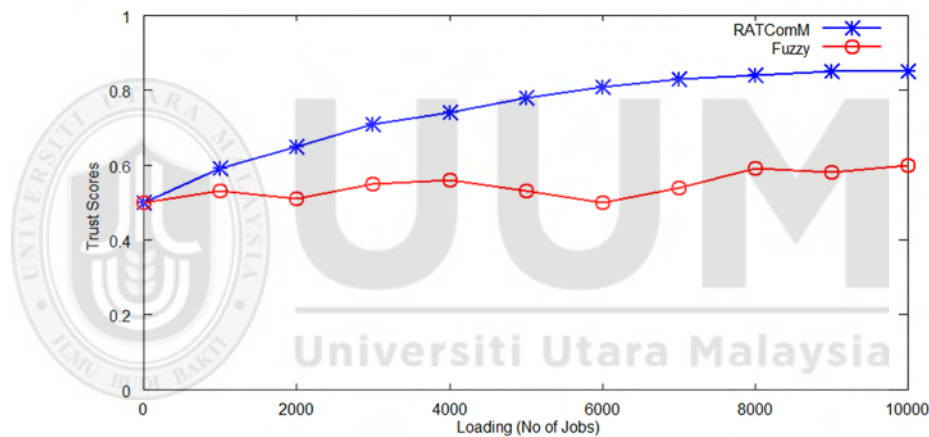


Figure 7.18. Trust Scores Computed based on Response Time

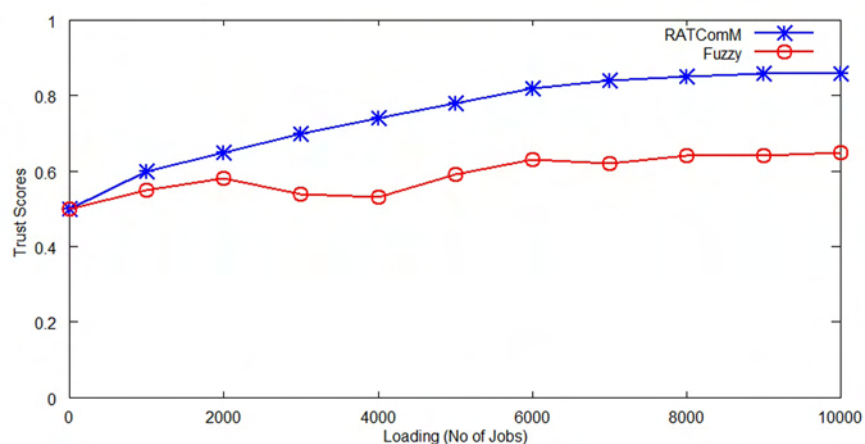


Figure 7.19. Trust Scores Computed based on Service Time

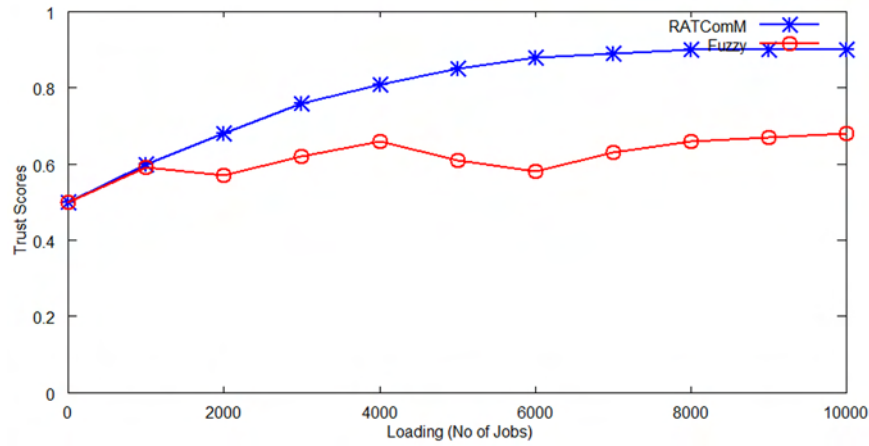


Figure 7.20. Trust Scores Computed based on Availability

The above figures show that the RATComM mechanism has better and stable performance than the fuzzy theory based trust computing mechanism. This is mainly due to the elimination of the effect of small fluctuations on the output through input validation and stability of output provided by the hysteresis function.

7.4.5 Performance Analysis of MuDTComM

The same environment has been used to evaluate the Multi-Dimensional Trust Computing Mechanism (MuDTComM). Figures 7.21 and 7.22 present the trust scores calculated with the proposed MuDTComM and fuzzy theory based mechanisms based on the combination of response time, service time and availability respectively. Though the fuzzy theory based mechanism was originally proposed for a single input function, it was slightly modified to accept multiple inputs. The inputs were combined into a single factor as explained in Chapter Four, before applying them to the trust computing mechanisms. This way both these mechanisms have been brought to the similar experimental environments, so that the outputs can be compared.

Figure 7.21 shows the trust scores computed using MuDTComM against that of fuzzy theory based trust computing mechanism. MuDTComM was used to compute trust scores at 90% and 95% confidence levels and equal weights were applied to all three

performance parameters, namely response time, service time and availability.

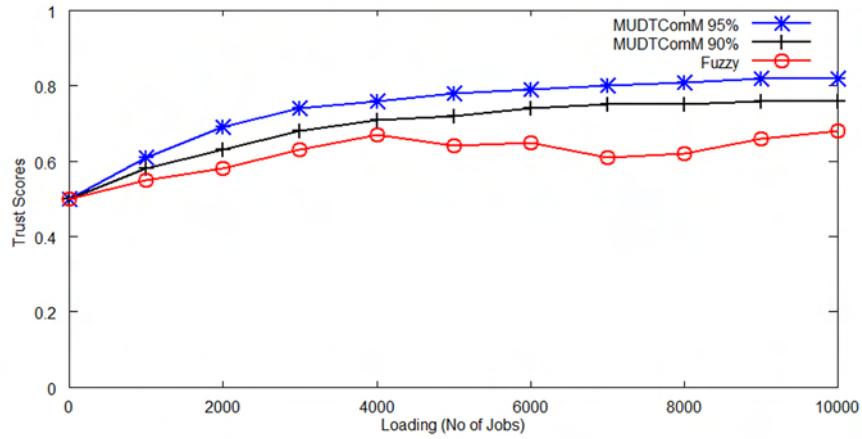


Figure 7.21. Trust Scores Computed by MuDTComM and Fuzzy Mechanisms

The experiments were repeated using different weighting factors for the service quality parameters. Figure 7.22 shows the trust scores computed using MuDTComM and fuzzy theory based mechanisms when response time, service time and availability were applied $w_1 = 0.5$, $w_2 = 0.3$ and $w_3 = 0.2$ respectively.

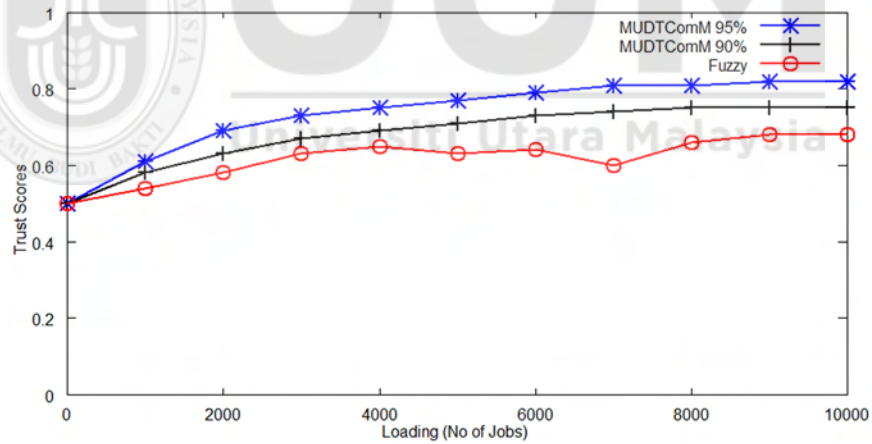


Figure 7.22. Effect of Weights on Trust Scores - MuDTComM vs Fuzzy

From Figures 7.21 and 7.22, it can be observed that MuDTComM performs better than fuzzy theory based mechanism for both cases of 90 percent and 95 percent confidence levels. It should also be noted that the MuDTComM scores are more stable than that of fuzzy theory based mechanism. This is mainly due to the reason that small fluctuations in the performance is tolerated by the proposed MuDTComM as any value falling within the confidence interval is taken as acceptable performance irrespective

of minor deviations in the actual values. On the other hand, fuzzy theory based mechanism responds to even small changes in the performance modifying the trust scores. Thus, it can be concluded that the proposed MuDTComM performs better than fuzzy theory based mechanism and also gives the freedom for customers to select their own confidence level for computing the trust scores. This enables the differentiation of customers based on the stringency of their service quality requirements.

7.5 Performance Analysis of Trust Distribution Mechanism

The simulation environment was slightly modified for the verification of the trust distribution mechanism. The simulation environment was setup to with ten data centers at different locations geographically distributed. These data centers were interconnected with each other using links with the speed of 4 Mbps. This interconnection speed was selected based on the world wide average Internet connection speed reported by OMICS International [250]. Each data center was designed to have only one server with equal capabilities as the main objective of these experiments was to compute the effect of the intermediate network on the trust scores. The configuration of a data center is as given in Table 7.6.

Table 7.6
Experiment Setup Attributes for the Evaluation of Trust Distribution Mechanism

Attribute	Sub-Attribute	Value
No. of servers		1
Server properties	Architecture	x86
	No. of processors	1 (Quad Core)
	Memory (RAM)	8 GB
	Storage	1 TB
	Network bandwidth	1 Gbps
VM allocation policy		Simple
VM scheduling method		Time and space shared
Cost		0

The performance requirements of the clients were then set on the service quality

parameters of response time, service time and availability and other resource requirements. Table 7.7 shows the client requirements set in for these experiments.

Table 7.7

Service Quality Requirements for Trust Distribution Mechanism

Parameter	Sub-Parameter	Value
Response time		500 ms
Service time		10 minutes
Availability		100%
Computing Power	Processor	2.27 GHz Clock speed
		6 MB Cache
		533 MHz Bus speed
	Memory	2 GB
Network speed	Storage	100 GB
	Bandwidth	100 Mbps
	Latency	3.33 ms

Each experiment was repeated ten (10) times to obtain stable results and their mean values were considered for validation. The results were validated by comparing them with that of the Super-Agent-based Framework for Reputation Management and Community Formation in Decentralized Systems proposed by Wang et al. in [51]. The above mechanism consists of two types of agents known as consumer agents and super agents for computing and interchanging reputation scores. The roles of consumer agents and super agents are assigned considering the capabilities of the hosts. The super agents have more capacity than consumer agents and they take additional responsibilities too. The super agents compute the reputation scores and share with other agents. Consumer agents use the reputation scores computed by super agents.

Super agent based scheme computes the trustworthiness (reputation) of a services s as given by Equation 7.4.

$$R_{sp}(s) = \frac{\sum_{i=1}^n T_c(sp_i) R_{sp_i}}{\sum_{i=1}^n T_c(sp_i)} \quad (7.3)$$

where

$R_{sp}(s)$ - reputation value for the service s

$T_c(sp_i)$ - the consumer agent's trust in the super-agent sp_i and

$R_{sp_i}(s)$ - reputation opinion provided by sp_i

and updated after every interaction by

$$\hat{T}_c(s) = \alpha T_c(s) + (1 - \alpha)e(s) \quad (7.4)$$

where

$T_c(s)$ and $\hat{T}_c(s)$ are the trust value of the service s before and after the update respectively

α is the learning rate - a fraction in the range of (0,1) and

$e(s)$ is the evaluation results of the interaction, where $e(s)$ will take either 1 or 0 depending on "satisfactory" or "not satisfactory" respectively.

The experiments consisted of 10 job submissions starting with 1000 jobs for the first submission and increasing the number of jobs per consecutive submission by another 1000 jobs. Thus the final job submission contained 10,000 jobs altogether cumulatively. Each job submission will create a single virtual machine and submit all the 1000 jobs to that virtual machine. The processing of the jobs is simulated by creating a cloudlet and running it for a predetermined time of 10 minutes as stated above. The availability was measured using the statistic that how many cloudlet instantiation succeeded on the first attempt itself.

An extensive set of experiments were carried out using response time as the service quality parameter. The initial trust was assumed to be 0.8 a value that was approached in the previous experiments as final score. The trust scores were then computed using PTDiMech and super agent based mechanism for different loading conditions. Figure 7.23 shows the trust scores computed using response time as input.

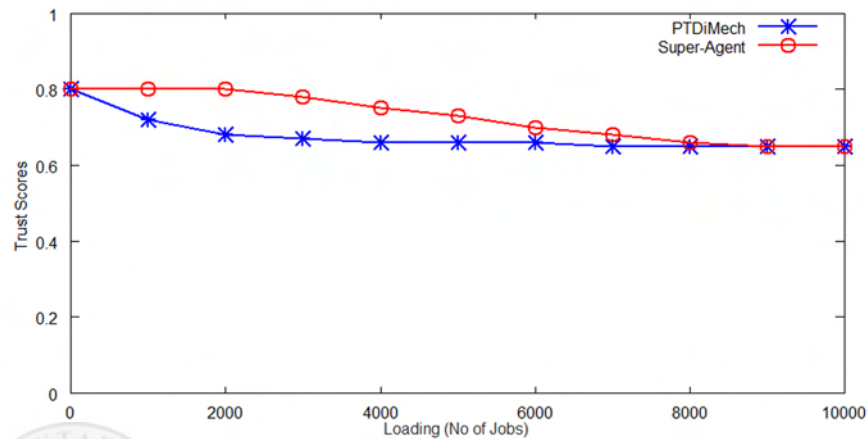


Figure 7.23. Trust Scores Computed based on Response Time

From Figure 7.23, it could be observed that both PTDiMech and super agent based mechanism settle down to a stable value. On the other hand, the proposed PTDiMech reaches that value much faster than the super agent based mechanism. The approaches taken by the respective mechanisms for adjusting the trust scores makes the difference in the leading to the final stable value. PTDiMech approaches the final score faster as from the beginning it computes the probability of meeting the service quality commitments. Hence it decays faster at the beginning followed by a slower rate while approaching the final score. On the other hand, the super-agent based mechanism gives the higher weight for the opinion of the super agent from which it receives the initial trust score. The opinion of the super agent stay stronger until the consumer agent obtains sufficient experience. This results in the slower approach rate. Hence the proposed PTDiMech works better and faster than the super agent based trust distribution mechanism.

Figure 7.24 shows the trust scores computed using service time as the parameter.

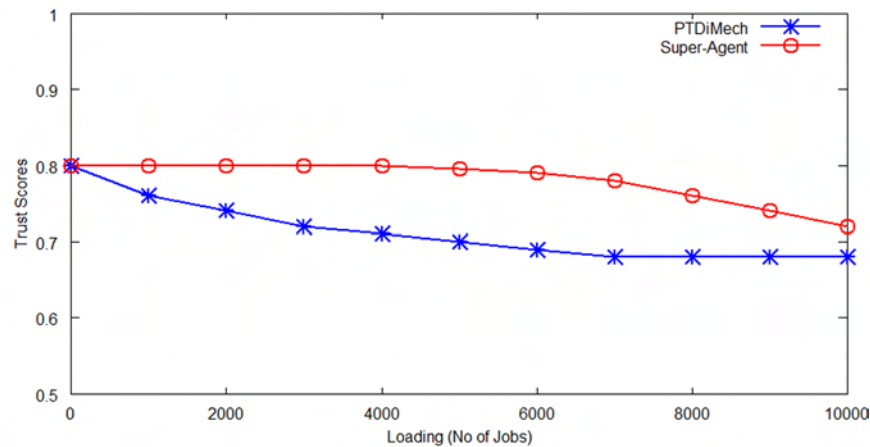


Figure 7.24. Trust Scores Computed based on Service Time

Similar to the case of response time, the trust scores computed using service time also shows similar trends. Figure 7.24 shows that the PTDiMech approaches the final score much faster than the super agent based mechanism. In this case, the super agent based mechanism acts comparatively very slowly. Obviously, the super agent based mechanism has not reached the final score yet, while the PTDiMech has already reached the stable final score. This is due to the reason that the service time is generally independent of the network dynamics and depends mainly on the capabilities of the server. Hence the opinion of the super agent supplying the initial trust score influences the resulting trust score for a long time until the client (recipient monitor) gains sufficient experience to modify the trust scores.

Figure 7.25 shows the trust scores computed using availability as input service parameter. From this figure, it can be seen that the proposed PTDiMech approaches the final score much faster than the super agent based mechanism.

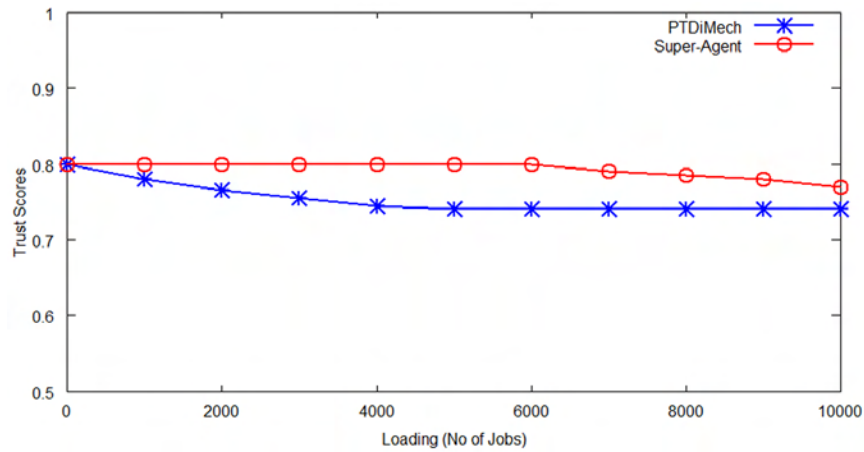


Figure 7.25. Trust Scores Computed based on Availability

Figures 7.26 and 7.27 show the performance of PTDiMech against that of super agent based mechanism for the combined trust scores based on multiple performance parameters. Similar to other experiments, the combined trust scores were also computed by combining the performance parameters response time, service time and availability. Figure 7.26 shows the trust scores computed by combining the performance parameters using equal weights.

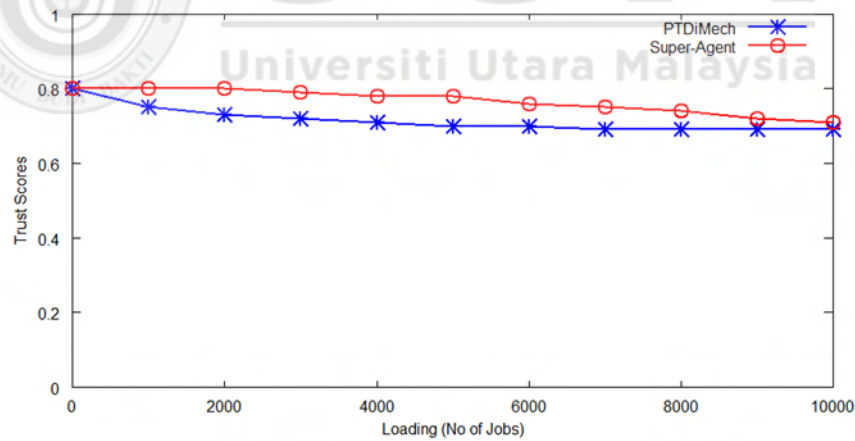


Figure 7.26. Trust Scores Computed with Equal Weights to All Parameters

Figure 7.27 shows the trust scores computed by combining the performance parameters using different weights. The weights applied to response time, service time and availability are $w_1 = 0.5$, $w_2 = 0.3$ and $w_3 = 0.2$ respectively.

From Figures 7.26 and 7.27, it can be seen that the PTDiMech reaches the final stable

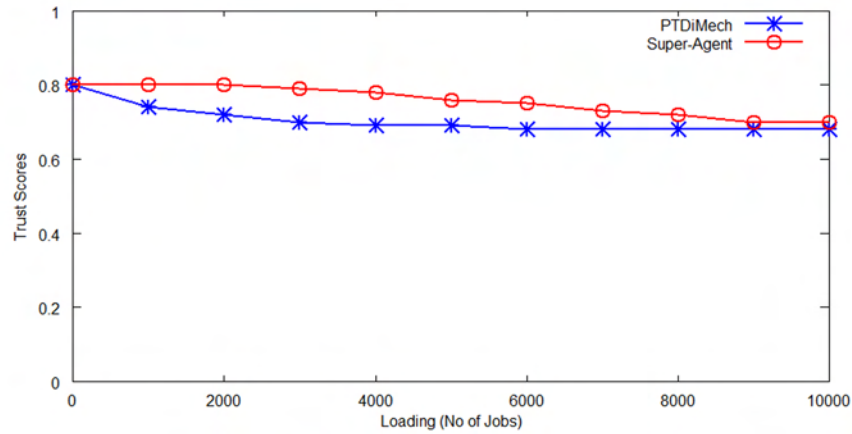


Figure 7.27. Trust Scores Computed with Different Weights to Parameters

trust score for the remote system faster than super agent based mechanism for both cases. This is similar to the performance of the trust distribution mechanisms, when the different service quality parameters were considered separately to compute the trust scores.

Under all the conditions, the PTDiMech mechanism proposed in this research reaches the final stable trust score faster with fewer interactions than the super agent based mechanism. This observation is true irrespective of the individual parameter selected or they are combined together. Hence it can be concluded that the proposed PTDiMech works better in distributing the trust scores among the cooperating trust monitors than the super agent based mechanism. Hence the customers would be able to identify the most suitable cloud service provider with relative ease with PTDiMech than the super agent based mechanism. This would result in less SLA violations helping both the service provider as well as the customers.

7.6 Summary

This chapter presented the performance evaluation and validation of the proposed service quality quantification, trust computing and trust distribution mechanisms proposed in Chapters Four, Five and Six. Extensive experiments were carried out in a

simulated environment built with the CloudSim software. The simulation results show that the mechanisms proposed in this research work better than the mechanisms that were used to validate them. The mechanisms used for validation were selected from journal articles that were published recently. The following paragraphs summarize the main contributions of this research presented in this chapter.

The two service quality quantification mechanisms proposed in this research are SP-SQQM and MP-SQQM for computing the trust score using an identified single service quality parameter and multiple service quality parameters combined based on a predefined weighting scheme. These mechanisms were evaluated against Combined Trust Model and QoS-based Trust Model proposed in [48] and [49] respectively. Both the mechanisms proposed in this research outperform both CTM and QoS-based Trust Models in reaching the final scores faster with fewer interactions and providing a stable trust scores.

The two main trust computing mechanisms developed in this research are RATComM and MuDTComM that monitor the performance of cloud systems continuously and update the trust scores. These mechanisms were developed in step by step manner starting from a simple adaptive continuous trust evolution mechanism (ACTEM) proposed in the same research. Both RATComM and MuDTComM were evaluated against the fuzzy theory based trust computing mechanism proposed in [50] under rigorous conditions. Under all the conditions, both RATComM and MuDTComM perform better than the fuzzy theory based mechanism in terms of reaching the final score faster and providing a stable trust score.

Finally PTDiMech was proposed for distributing trust scores between cooperating trust computing nodes. Distribution of trust is necessary as the cloud systems are

deployed throughout the Internet covering large geographical areas. A single trust computing node cannot cover all the cloud systems distributed throughout the Internet. So far no such trust distribution system has been proposed for cloud computing. Hence the modified super agent based trust distribution mechanism proposed for decentralized systems was selected for evaluating the performance of PTDiMech. The experiments conducted for evaluating the trust distribution mechanisms show that PTDiMech reaches the stable final trust score for the remote system faster than super agent based mechanism.



CHAPTER EIGHT

CONCLUSIONS AND FUTURE WORK

8.1 Introduction

This research was aimed at developing an adaptive service quality measuring and quantification mechanism for cloud computing. The previous chapters elaborated the research conducted and the mechanisms developed in detail along with the detailed description on the literature review carried out and the methodology adopted. This chapter presents the conclusion of the research in terms of highlighting the important points, contributions, limitations of the work and recommendations for future work.

The organization of this chapter is as follows. Section 8.1 provides the brief introduction to the chapter along with its organization. Section 8.2 summarizes the work carried out highlighting the important points. Section 8.3 presents the contributions of this research, while Section 8.4 explains the limitations of the work. Finally Section 8.5 offers some suggestion for future work that provides an indication to the directions in which this work can be further developed.

8.2 Summary of Research

Cloud computing has been the newest paradigm in computing that delivers the computing resources such as processing power, development platforms and software applications over the Internet as services. Cloud computing benefits both the service providers as well as customers at the same time. Service providers derive their benefits through the improved utilization and productivity of the resources while the customers benefits come from reduced cost and being charged for only the actual usage of resources. When the computing requirements are outsourced to a cloud service, it has the capability to closely follow the demands of the customers without over committing

nor under provisioning the required resources. This concept is commonly known as the elastic property of cloud computing.

The enable technology of cloud computing is virtualization. Virtualization enables the creation of temporary computers on the fly equipped with the right amount of hardware resources such as computing power, memory, storage and network capacities. These on the fly virtual computers are hosted on top of the real hardware only for the duration of the use and then removed releasing all the resources for another virtual computer that becomes active later. This provides the service providers the freedom of selling the same resources to multiple customers increasing their utilization. Such increased utilization of resources results in the reduction of per user cost benefiting both users and providers.

The virtualization helps loading of multiple virtual machines concurrently on the same hardware. The required isolation and security between the concurrently loaded virtual machines is provided by the virtual machine manager. Though these virtual machines are isolated from one another by the virtual machine manager, they have to share many common resources such as the processing cores and system bus. This puts a limitation on the number of virtual machines to be loaded on given system depending on the capacity of real resources [165]. Loading beyond this number would degrade performance of virtual machines affecting all the customers.

The customers and service providers enter into a service level agreement at the inception. This agreement stipulates all the conditions to be met by both parties along with any penalties to paid, if any when these conditions are violated. Though the service level agreement provides a written guarantee for the customers on the performance of the service provider, it would be better for the customers to know

the capabilities of the providers given the requirements of the customers beforehand.

This research concentrates on filling this gap of monitoring the service quality of cloud service providers and ranking them based on their performance. In this regard, this research developed mechanisms that could objectively quantify the service quality received by the customers. These mechanisms are categorized into three main groups. They are namely, service quality quantification mechanisms, trust computing mechanisms and trust distribution mechanism.

The service quality quantification mechanisms developed in this research are capable of measuring the performance using either a single parameter or multiple parameters. Though the mechanisms developed have been tested for a given set of parameters, they can be applied with any parameter without modifications. Hence users not only have the freedom to select the parameters but also can determine relative importance of them too.

The trust computing mechanisms enable the monitor continuously track the performance of the service provider and modify the performance score known as the trust score adaptively. This exposes the actual performance level of the service provider based on the actual loading of the system. The proposed mechanisms adjust the trust scores proactively taking only the real changes in performance into account. The temporary fluctuations in performances are identified and dealt with appropriately through a statistical validation of inputs. This makes these mechanisms rugged in the face of temporary fluctuations.

The third type of mechanism developed in this research exchanges the trust scored between cooperating monitors. The exchange of trust scores between cooperating

monitors enable the system cover a large geographical area helping the customers to identify suitable service providers from any part of the Internet. The proposed decentralized architecture of the trust distribution mechanism makes the system resilient avoiding single points of failures.

All the proposed mechanisms were tested for their functionality and validated by comparing with other mechanisms proposed in the literature using simulations. The simulation environment was setup in CloudSim, the most popular open source cloud computing simulator in the market. The simulation environment was setup with single or multiple data centers depending on the mechanism tested. The experiments carried out show that the proposed mechanisms outperform all the other mechanisms proposed in the literature.

Overall the proposed mechanisms would help the prospective cloud computing customers identify the right service provider, who could meet their service requirements. This will benefit both customers and service providers who can detect the service level degradations long before they become critical issues.

8.3 Research Contributions

The overall contribution of this research is the development of service quality monitoring, quantification and distribution mechanisms. The specific contributions of this research are listed below:

1. Service quality quantification mechanisms for cloud computing.
 - a) The mathematical formulation of service quality parameters.
 - b) The incorporation of Bayes' theorem into quantification of service quality.

- c) The development of single parameter service quality quantification mechanism.
 - d) The development of multi parameter service quality quantification mechanism.
 - e) The devising a method for incorporation of different priorities into the multi parameter service quality quantification mechanism.
2. Adaptive trust computing mechanisms for cloud computing.
- a) The development of adaptive continuous trust evolution mechanism.
 - b) The development of memoryless trust computing mechanism.
 - c) The development of robust adaptive trust computing mechanism.
 - d) The development of multi-dimensional trust computing mechanism.
3. Trust distribution mechanism for cloud computing.
- a) The incorporation of probability into computing the weighting factors for received trust scores.
 - b) The development of probability-based trust distribution mechanism.

8.4 Research Limitations

Although this research has successfully developed, tested and evaluated several mechanisms that can be collectively employed to track and quantify the service quality of cloud computing systems, it has only been tested under simulated conditions. Despite the simulation environment been setup to closely resemble with the real word conditions, all the uncertainties that can be experienced in the real world cannot be included into the simulation environment. Though it has theoretically been shown that the proposed multi-dimensional mechanisms can incorporate any service quality parameter, they have tested with only a few selected parameters. The real Internet hosts many different types of services. The limited resources such as bandwidth of

links and processing power of networking devices need to be shared by cloud as well as non cloud applications. In a simulated environment, it is not possible to measure the effects of non cloud applications on the proposed mechanisms or cloud applications.

8.5 Recommendations for Future Work

The proposed mechanisms can effectively quantify the service quality of cloud computing systems and rank them based on their performance. However there are some limitations and pending work that can be carried out as future work. The following are some suggestions that can be carried out in the future extending this work.

1. *Testing and evaluating the proposed mechanisms using testbeds and in the real cloud systems.*

The mechanisms were tested in a simulated environment only. This was also highlighted as the one of the limitations of this work. Hence it is suggested that these mechanisms must be tested in the real environment and their performance evaluated on a future date.

2. *Testing the mechanisms using some other service quality attributes.*

The simulation was limited to testing the service quality of IaaS services in the cloud. This is due to the limitations of the currently available cloud computing simulators. As future work, they can be tested for measuring the service quality of PaaS as well as SaaS services.

3. *Monitoring the performance of a heterogeneous cloud computing systems.*

The simulation environment was setup with only homogeneous cloud systems in order to create repeatable experiments. It is suggested that the environment is modified to include computing systems of different types in data center and the performance of the proposed mechanism are tested as future work.



REFERENCES

- [1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Journal of Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, June 2009.
- [2] F. Hu, M. Qiu, J. Li, T. Grant, D. Tylor, S. McCaleb, L. Butler, and R. Hamner, "A review on cloud computing: Design challenges in architecture and security," *Journal of Computing and Information Technology*, vol. 19, no. 1, pp. 25–55, 2011.
- [3] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing - the business perspective," *Decision Support Systems*, vol. 51, no. 1, pp. 176–189, 2011.
- [4] J. Jaatmaa, "Financial aspects of cloud computing business models," Master's thesis, School of Economics, Aalto University, Finland, 2010.
- [5] S. Islam, K. Lee, A. Fekete, and A. Liu, "How a consumer can measure elasticity for cloud platforms," in *3rd Joint WOSP/SIPEW International Conference on Performance Engineering*, Boston, MA, USA, 2012, pp. 85–96.
- [6] AWS. (2012) Capacity utilization curve. AWS Economics Center. - Accessed on 05/01/2012. [Online]. Available: <http://aws.amazon.com/economics/>
- [7] K. Collins, *Exploring Business*. Hoboken, NJ: Prentice Hall, 2007.
- [8] K. Rafique, A. W. Tareen, M. Saeed, J. Wu, and S. S. Qureshi, "Cloud computing economics opportunities and challenges," in *4th IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, Shenzhen, China, 2011, pp. 401–406.
- [9] R. M. Sharma, "The impact of virtualization in cloud computing," *International Journal of Recent Development in Engineering and Technology*, vol. 3, no. 1, pp. 197–202, 2014.
- [10] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," in *Second IEEE International Conference on Cloud Computing Technology and Science*, Indianapolis, IN, USA, 2010, pp. 127–134.
- [11] A. A. Semnanian, J. Pham, B. Englert, and X. Wu, "Virtualization technology and its impact on computer hardware architecture," in *Eighth International Conference on Information Technology: New Generations*, Las Vegas, NV, USA, 2011, pp. 719–724.
- [12] R. Y. Ameen and A. Y. Hamo, "Survey of server virtualization," *International Journal of Computer Science and Information Security*, vol. 11, no. 3, pp. 1–10, 2013.

- [13] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in *10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN)*, Kaohsiung, Taiwan, 2009, pp. 4–16.
- [14] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," in *International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, 2010, pp. 6–20.
- [15] Y. Li, W. Li, and C. Jiang, "A survey of virtual machine system: Current technology and future trends," in *Third International Symposium on Electronic Commerce and Security*, Guangzhou, China, 2010, pp. 332–336.
- [16] A. Apostu, F. Puican, G. Ularu, G. Suciu, and G. Todoran, *Recent Advances in Applied Computer Science and Digital Services*. New York, NY: WSEAS Press, 2013, ch. Study on Advantages and Disadvantages of Cloud Computing - The Advantages of Telemetry Applications in the Cloud, pp. 118–123.
- [17] A. Thilakarathne and J. I. Wijayanayake, "Security challenges of cloud computing," *International Journal of Scientific and Technology Research*, vol. 3, no. 11, pp. 200–203, 2014.
- [18] A. A. Omotunde, O. Awodele, S. O. Kuyoro, and C. Ajaegbu, "Survey of cloud computing issues at implementation level," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 4, no. 1, pp. 91–96, 2013.
- [19] D. C. Marinescu, *Cloud Computing: Theory and Practice*, 1st ed. Morgan Kaufmann, 2013.
- [20] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 931–945, 2011.
- [21] G. Feng, S. Garg, R. Buyya, and W. Li, "Revenue maximization using adaptive resource provisioning in cloud computing environments," in *13th ACM/IEEE International Conference on Grid Computing*, Beijing, China, 2012, pp. 192–200.
- [22] *Amazon S3 Service Level Agreement*, accessed on February 07, 2013. [Online]. Available: <http://aws.amazon.com/s3-sla/>
- [23] H. A. Akpan and B. R. J. Vadhanam, "A survey on quality of service in cloud computing," *International Journal of Computer Trends and Technology*, vol. 27, no. 1, pp. 58–63, 2015.
- [24] H. Blodget, *Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data*, accessed on February 07, 2013. [Online]. Available: <http://www.businessinsider.com/amazon-lost-data-2011-4>

- [25] S. P. Ahuja and S. Mani, "Availability of services in the era of cloud computing," *Network and Communication Technologies*, vol. 1, no. 1, pp. 2–6, 2012.
- [26] Y. Sun, J. Zhang, Y. Xiong, and G. Zhu, "Data security and privacy in cloud computing," *International Journal of Distributed Sensor Networks*, pp. 1–9, 2014.
- [27] K. Xiong and H. Perros, "Service performance and analysis in cloud computing," in *IEEE World Congress on Services, Los Angeles, CA, 2009*, pp. 693–700.
- [28] G. Singh and V. K. Sachdeva, "Impact and challenges of cloud computing in current scenario," *International Journal of Social Science & Interdisciplinary Research*, vol. 1, no. 10, pp. 131–144, 2012.
- [29] K. Alhamazani, R. Ranjan, P. P. Jayaraman, K. Mitra, C. Liu, F. Rabhi, D. Georgakopoulos, and L. Wang, "Cross-layer multi-cloud real-time application QoS monitoring and Benchmarking As-a-Service framework," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–13, 2015.
- [30] P. A. Salz, "Monitoring mobile application performance," *Journal of Direct, Data and Digital Marketing Practice*, vol. 15, pp. 219–221, 2014.
- [31] S. Bosse, C. Schulz, and K. Turowski, "Predicting availability and response times of IT services," in *22nd European Conference on Information Systems, Tel Aviv, Israel, 2014*, pp. 1–14.
- [32] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, pp. 1012–1023, 2013.
- [33] J. S. Ward and A. Barker, "Observing the clouds: A survey and taxonomy of cloud monitoring," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 3, no. 24, pp. 1–30, 2014.
- [34] M. H. Mohamaddiah, A. Abdullah, S. Subramaniam, and M. Hussin, "A survey on resource allocation and monitoring in cloud computing," *International Journal of Machine Learning and Computing*, Vol. 4, No. 1, February 2014, vol. 4, no. 1, pp. 31–38, 2014.
- [35] A. Brinkmann, C. Fiehe, A. Litvina, I. Luck, L. Nagel, K. Narayanan, F. Ostermair, and W. Thronicke, "Scalable monitoring system for clouds," *IEEE/ACM 6th International Conference on Utility and Cloud Computing, Dresden, Germany*, pp. 351–356, 2013.
- [36] S. Suakanto, S. H. Supangkat, Suhardi, and R. Saragih, "Performance measurement of cloud computing services," *International Journal on Cloud Computing: Services and Architecture*, vol. 2, no. 2, pp. 9–20, 2012.
- [37] G. Aceto, A. Botta, W. de Donato, and A. Pescapé, "Cloud monitoring: Definitions, issues and future directions," *1st IEEE International Conference on Cloud Networking, Paris, France*, pp. 63–67, 2012.

- [38] R. Buyya, S. K. Garg, and R. N. Calheiros, "SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions," in *International Conference on Cloud and Service Computing, Hong Kong*, 2011, pp. 1–10.
- [39] W. N. Chen and J. Zhang, "A set-based discrete pso for cloud workflow scheduling with user-defined qos constraints," in *IEEE International Conference on Systems, Man and Cybernetics, Seoul, Korea*, 2012, pp. 773–778.
- [40] R. V. den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds," in *Third IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Athens, Greece*, 2011, pp. 320–327.
- [41] V. C. Emeakaro, I. Brandic, M. Maurer, and I. Breskovic, "Sla-aware application deployment and resource allocation in clouds," in *35th IEEE Annual Computer Software and Applications Conference Workshops (COMPSACW)*, 2011, pp. 298–303.
- [42] K. Alhamazani, R. Ranjan, F. Rabhi, L. Wang, and K. Mitra, "Cloud monitoring for optimizing the QoS of hosted applications," in *4th IEEE International Conference on Cloud Computing Technology and Science, Taipei, Taiwan*, 2012, pp. 765–770.
- [43] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, Austin, TX*, 2008, pp. 1–10.
- [44] R. Kaur, "A review of computing technologies: Distributed, utility, cluster, grid and cloud computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 5, no. 2, pp. 144–148, 2015.
- [45] R. Buyya, R. N. Calheiros, and X. Li, "Autonomic cloud computing: Open challenges and architectural elements," in *3rd International Conference on Emerging Applications of Information Technology, Kolkata, India*, 2012, pp. 3–10.
- [46] S. Kermally, *Management Ideas in Brief*. Oxford, UK: Butterworth-Heinemann, 1997.
- [47] M. Boniface, B. Nasser, J. Papay, S. C. Phillips, A. Servin, X. Yang, Z. Zlatev, S. V. Gogouvitis, G. Katsaros, K. Konstanteli, G. Kousiouris, A. Menychtas, and D. Kyriazis, "Platform-as-a-Service architecture for real-time quality of service management in clouds," in *Fifth International Conference on Internet and Web Applications and Services, Barcelona, Spain*, 2010, pp. 155–160.
- [48] P. D. Manuel, S. T. Selvi, and M. I. Abd-El Barr, "A novel trust management system for cloud computing - IaaS providers," *Journal of Combinatorial Mathematics and Combinatorial Computing*, vol. 79, pp. 3–22, 2011.

- [49] P. Manuel, "A trust model of cloud computing based on quality of service," *Annals of Operations Research*, pp. 1–12, 2013.
- [50] L. Gu, J. Zhong, C. Wang, Z. Ni, and Y. Zhang, "Trust model in cloud computing environment based on fuzzy theory," *International Journal of Computers Communications and Control*, vol. 9, no. 5, pp. 570–583, 2014.
- [51] Y. Wang, J. Zhang, and J. Vassileva, "Super agent based framework for reputation management and community formation in decentralized systems," *Computational Intelligence*, pp. 1–26, 2014.
- [52] R. Prodan and S. Ostermann, "A survey and taxonomy of Infrastructure as a Service and web hosting cloud providers," in *10th IEEE/ACM International Conference on Grid Computing, Banff, Alberta, Canada*, 2009, pp. 17–25.
- [53] N. R. G. Charan, S. T. Rao, and P. V. S. Srinivas, "Deploying an application on the cloud," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 5, pp. 119–125, 2011.
- [54] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of several cloud computing platforms," in *Second International Symposium on Information Science and Engineering, Beijing, China*, 2009, pp. 23–27.
- [55] M. Zhou, R. Zhang, D. Zeng, and W. Qian, "Services in the cloud computing era: A survey," in *Fourth International Universal Communication Symposium, Beijing, China*, 2010, pp. 40–46.
- [56] J. Dykstra and A. T. Sherman, "Acquiring forensic evidence from Infrastructure-as-a-Service cloud computing: Exploring and evaluating tools, trust, and techniques," *Digital Investigation*, vol. 9, pp. S90–S98, 2012.
- [57] K. Hashizume, E. B. Fernandez, and M. M. Larrondo-Petrie, "Cloud service model patterns," in *19th Conference Pattern Languages of Programs, Tucson, AZ, USA*, 2012.
- [58] G. Kulkarni, P. Khatawkar, and J. Gambhir, "Cloud computing-platform as service," *International Journal of Engineering and Advanced Technology*, vol. 1, no. 2, pp. 115–120, 2011.
- [59] J. Kong, "Protecting the confidentiality of virtual machines against untrusted host," in *International Symposium on Intelligence Information Processing and Trusted Computing (IPTC), Huanggang, China*, 2010, pp. 364–368.
- [60] M. A. Bamiah and S. N. Brohi, "Exploring the cloud deployment and service delivery models," *International Journal of Research and Reviews in Information Sciences*, vol. 1, no. 3, pp. 77–80, 2011.
- [61] L. Savu, "Cloud computing: Deployment models, delivery models, risks and research challenges," in *International Conference on Computer and Management, Wuhan, China*, 2011, pp. 1–4.

- [62] L. Wang, J. Zhan, W. Shi, and Y. Liang, "In cloud, can scientific communities benefit from the economies of scale?" *IEEE Transactions on Parallel And Distributed Systems*, vol. 23, no. 2, pp. 296–303, 2012.
- [63] D. Velez and P. Zlateva, *Open Research Problems in Network Security*, ser. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2011, vol. 6555, ch. Cloud Infrastructure Security, pp. 140–148.
- [64] J. M. Pedersen, M. T. Riaz, J. C. Junior, B. Dubalski, D. Ledzinski, and A. Patel, "Assessing measurements of QoS for global cloud computing services," in *9th IEEE International Conference on Dependable, Autonomic and Secure Computing, Sydney, Australia*, 2011, pp. 682–689.
- [65] X. Liu, Y. Yang, D. Yuan, G. Zhang, W. Li, and D. Cao, "A generic QoS framework for cloud workflow systems," in *9th IEEE International Conference on Dependable, Autonomic and Secure Computing, Sydney, Australia*, 2011, pp. 713–720.
- [66] W. Li, Q. Zhang, J. Wu, J. Li, and H. Zhao, "Trust-based and QoS demand clustering analysis customizable cloud workflow scheduling strategies," in *IEEE International Conference on Cluster Computing Workshops, Beijing, China*, 2012, pp. 111–119.
- [67] M. S. Mushtaq, B. Augustin, and A. Mellouk, "Empirical study based on machine learning approach to assess the QoS/QoE correlation," in *17th European Conference on Networks and Optical Communications, Vilanova i la Geltru, Spain*, 2012, pp. 1–7.
- [68] J. Ma, R. Sun, and A. Abraham, "Toward a lightweight framework for monitoring public clouds," in *Fourth International Conference on Computational Aspects of Social Networks, Sao Carlos, Brazil*, 2012, pp. 361–365.
- [69] Q. Zhu and G. Agrawal, "Resource provisioning with budget constraints for adaptive applications in cloud environments," *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 497–511, 2012.
- [70] J. Li, Q. Wang, D. Jayasinghe, S. Malkowski, P. Xiong, C. Pu, Y. Kanemasa, and M. Kawaba, "Profit-based experimental analysis of iaas cloud performance: Impact of software resource allocation," in *Ninth IEEE International Conference on Services Computing, Honolulu, HI, USA*, 2012, pp. 344–351.
- [71] B. Sharma, R. K. Thulasiram, P. Thulasiraman, S. K. Garg, and R. Buyya, "Pricing cloud compute commodities: A novel financial economic model," in *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Ottawa, ON, Canada*, 2012, pp. 451–457.
- [72] D. Adami, C. Callegari, S. Giordano, and M. Pagano, "A hybrid multidimensional algorithm for network-aware resource scheduling in clouds and grids," in *IEEE International Conference on Communications, Ottawa, ON, Canada*, 2012, pp. 1297–1301.

- [73] S. C. Phillips, V. Engen, and J. Papay, "Snow white clouds and the seven dwarfs," in *Third IEEE International Conference on Cloud Computing Technology and Science, Athens, Greece*, 2011, pp. 738–745.
- [74] A. Gohad, K. Ponnalagu, and N. C. Narendra, "Model driven provisioning in multi-tenant clouds," in *Annual Service Research and Innovation Institute Global Conference*, 2012, pp. 11–20.
- [75] G. N. Iyer and B. Veeravalli, "On the resource allocation and pricing strategies in compute clouds using bargaining approaches," in *17th IEEE International Conference on Networks, Singapore*, 2011, pp. 147–152.
- [76] S. P. Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, "Markovian workload characterization for QoS prediction in the cloud," in *IEEE International Conference on Cloud Computing, Washington, DC, USA*, 2011, pp. 147–154.
- [77] Y. Kouki, T. Ledoux, and R. Sharrock, "Cross-layer SLA selection for cloud services," in *First International Symposium on Network Cloud Computing and Applications, Toulouse, France*, 2011, pp. 143–147.
- [78] L. Wu, S. K. Garg, and R. Buyya, "SLA-based resource allocation for Software as a Service provider (SaaS) in cloud computing environments," in *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Newport Beach, CA, USA*, 2011, pp. 195–204.
- [79] T. Chauhan, S. Chaudhary, V. Kumar, and M. Bhise, "Service level agreement parameter matching in cloud computing," in *World Congress on Information and Communication Technologies, Mumbai, India*, 2011, pp. 564–570.
- [80] R. S. Padilla, S. K. Milton, and L. W. Johnson, "Components of service value in business-tobusiness cloud computing," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 4, no. 15, pp. 1–20, 2015.
- [81] S. V. Gogouvitis, V. Alexandrou, N. Mavrogeorgi, S. Koutsoutos, D. Kyriazis, and T. Varvarigou, "A monitoring mechanism for storage clouds," in *Second International Conference on Cloud and Green Computing, Xiangtan, Hunan, China*, 2012, pp. 153–159.
- [82] F. Stoicuta, I. Ivanciu, E. Minzat, A. B. Rus, and V. Dobrota, "An opennetinf-based cloud computing solution for cross-layer qos: Monitoring part using ios terminals," in *10th International Symposium on Electronics and Telecommunications, Timisoara, Romania*, 2012, pp. 167–170.
- [83] K. Saravanan and M. L. Kantham, "An enhanced QoS architecture based framework for ranking of cloud services," *International Journal of Engineering Trends and Technology*, vol. 4, no. 4, pp. 1022–1031, 2013.
- [84] N. Kumar and S. Agarwal, "QoS based cloud service provider selection framework," *Research Journal of Recent Sciences*, vol. 3, pp. 7–12, 2014.

- [85] M. K. Goyal, A. Aggarwal, P. Gupta, and P. Kumar, "QoS based trust management model for cloud IaaS," in *2nd IEEE International Conference on Parallel, Distributed and Grid Computing, Solan, India*, 2012, pp. 843–847.
- [86] R. F. AlCattan, "Integration of cloud computing and web 2.0 collaboration technologies in e-learning," *International Journal of Computer Trends and Technology*, vol. 12, no. 1, pp. 46–55, 2014.
- [87] N. Phaphoom, X. Wang, and P. Abrahamsson, "Foundations and technological landscape of cloud computing," *ISRN Software Engineering*, vol. 2013, pp. 1–31, 2013.
- [88] R. Kaur, "Cloud computing," *International Journal of Computer Science and Technology*, vol. 2, no. 3, pp. 373–381, 2011.
- [89] A. Abdelmaboud, D. N. A. Jawawi, I. Ghani, A. Elsafi, and B. Kitchenham, "Quality of service approaches in cloud computing," *Journal of Systems and Software*, vol. 101, no. C, pp. 159–179, 2015.
- [90] G. V. Bochmann, B. Kerherve, H. Lutfiyya, M. V. M. Salem, and H. Ye, "Introducing QoS to electronic commerce applications," in *2nd International Symposium on Electronic Commerce (ISEC), Hong Kong*, 2001, pp. 138–147.
- [91] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya, "SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter," in *11th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2011), Melbourne, Australia*, ser. LNCS 7016, vol. I. Springer, 2011, pp. 371–384.
- [92] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguade, "Enabling resource sharing between transactional and batch workloads using dynamic application placement," in *9th ACM/IFIP/USENIX International Conference on Middleware. Leuven, Belgium*, 2008, pp. 203–222.
- [93] N. Thio and S. Karunasekera, "Automatic measurement of a QoS metric for web service recommendation," in *Australian Software Engineering Conference, Brisbane, Australia*, 2005, pp. 202–211.
- [94] G. E. Goncalves, P. T. Endo, T. D. Cordeiro, A. V. de Almeida Palhares, D. Sadok, J. Kelner, B. Melander, and J. E. Mangs, "Resource allocation in clouds: Concepts, tools and research challenges," in *29th Brazilian Symposium on Computer Networks and Distributed Systems, Campo Grande, MS, Brazil*, 2011, pp. 197–240.
- [95] S. Mulay and S. Jain, "Enhanced equally distributed load balancing algorithm for cloud computing," *International Journal of Research in Engineering and Technology*, vol. 2, no. 6, pp. 976–980, 2013.
- [96] S. Begum and C. S. R. Prashanth, "Review of load balancing in cloud computing," *International Journal of Computer Science Issues*, vol. 10, no. 1, pp. 343–352, 2013.

- [97] T. K. Chiew and K. Renaud, "Disassembling web site response time," in *12th European Conference on Information Technology Evaluation, Turku, Finland*, 2005, pp. 137–145.
- [98] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of Amazon EC2 data center," in *IEEE INFOCOM, San Diego, CA*, 2010, pp. 1–9.
- [99] M. Alhamad, T. Dillon, C. Wu, and E. Chang, "Response time for cloud computing providers," in *12th International Conference on Information Integration and Web-based Applications & Services, Paris, France*, 2010, pp. 603–606.
- [100] D. Ardagna, G. Casale, M. Ciavotta, J. F. Perez, and W. Wang, "Quality-of-service in cloud computing: Modeling techniques and their applications," *Journal of Internet Services and Applications*, vol. 5, no. 11, pp. 1–17, 2014.
- [101] P. Cremonesi, K. Dhyani, and A. Sansottera, "Service time estimation with a refinement enhanced hybrid clustering algorithm," in *17th International Conference on Analytical and Stochastic Modeling Techniques and Applications, Cardiff, UK*, 2010, pp. 291–305.
- [102] A. I. El-Nashar, "To parallelize or not to parallelize, speed up issue," *International Journal of Distributed and Parallel Systems*, vol. 2, no. 2, pp. 14–28, 2011.
- [103] C. Wang, L. Xing, H. Wang, Z. Zhang, and Y. Dai, "Processing time analysis of cloud services with retrying fault-tolerance technique," in *First IEEE International Conference on Communications in China, Beijing, China*, 2012, pp. 63–67.
- [104] F. S. Ahmed, A. Aslam, S. Ahmed, and M. A. Q. Bilal, "Comparative study of scalability and availability in cloud and utility computing," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 2, no. 12, pp. 705–713, 2011.
- [105] L. Zhao and K. Sakurai, "Improving cost-efficiency through failure-aware server management and scheduling in cloud," in *2nd International Conference on Cloud Computing and Services Science, Porto, Portugal*, 2013, pp. 23–38.
- [106] M. G. Avram, "Advantages and challenges of adopting cloud computing from an enterprise perspective," in *7th International Conference Interdisciplinarity in Engineering, Mures, Romania*, 2013, pp. 529–534.
- [107] Y. S. Dai, B. Yang, J. Dongarra, and G. Zhang, "Cloud service reliability: Modeling and analysis," in *15th IEEE Pacific Rim International Symposium on Dependable Computing, Shanghai, China*, 2009, pp. 1–17.
- [108] N. Yadav, V. B. Singh, and M. Kumari, "Generalized reliability model for cloud computing," *International Journal of Computer Applications*, vol. 88, no. 14, pp. 13–16, 2014.

- [109] T. Lynn, P. Healy, R. McClatchey, J. Morrison, C. Pahl, and B. Lee, “The case for cloud service trustmarks and Assurance-as-a-Service,” in *3rd International Conference on Cloud Computing and Services Science, Aachen, Germany*, 2013, pp. 1–6.
- [110] R. Scandariato, Y. Ofek, P. Falcarin, and M. Baldi, “Application-oriented trust in distributed computing,” in *Third International Conference on Availability, Reliability and Security, Barcelona, Spain*, 2008, pp. 434–439.
- [111] N. A. Al-Saiyd and N. Sail, “Data integrity in cloud computing security,” *Journal of Theoretical and Applied Information Technology*, vol. 58, no. 3, pp. 570–581, 2013.
- [112] H. Yu, Z. Shen, C. Miao, C. Leung, and D. Niyato, “A survey of trust and reputation management systems in wireless communications,” *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1755–1772, 2010.
- [113] Z. Gan, J. He, and Q. Ding, “Trust relationship modelling in e-commerce-based social network,” in *International conference on computational intelligence and security, Beijing, China*, 2009, pp. 206–210.
- [114] D. H. McKnight and N. L. Chervany, “Conceptualizing trust: A typology and e-commerce customer relationships model,” in *34th Hawaii International Conference on System Sciences, Island of Maui, HI, USA*, 2001.
- [115] W. Wang and G. S. Zeng, “Bayesian cognitive trust model based self-clustering algorithm for MANETs,” *Science China Information Sciences*, vol. 53, no. 3, pp. 494–505, 2010.
- [116] M. Gomez, J. Carbo, and E. C. Benac, “A cognitive trust and reputation model for the ART testbed,” *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial (in English)*, vol. 12, no. 39, pp. 29–40, 2008.
- [117] H. Quan and J. Wu, “CATM: A cognitive-inspired agent-centric trust model for online social networks,” in *Ninth Annual IEEE International Conference on Pervasive Computing and Communications, Seattle, WA, USA*, 2011.
- [118] C. Castelfranchi, R. Falcone, and G. Pezzulo, “Trust in information sources as a source for trust: A fuzzy approach,” in *Second International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne, Australia*, 2003, pp. 89–96.
- [119] S. de Paoli, G. R. Gangadharan, A. Kerr, V. D. Andrea, M. Serrano, and D. Botvich, “Toward trust as result: An interdisciplinary approach,” *Sprouts: Working Papers on Information Systems*, vol. 10, no. 8, pp. 1–6, 2010.
- [120] M. Akhoondi, J. Habibi, and M. Sayyadi, “Towards a model for inferring trust in heterogeneous social networks,” in *Second Asia International Conference on Modelling & Simulation, Kuala Lumpur, Malaysia*, 2008, pp. 52–58.
- [121] R. A. Menkes, “An economic analysis of trust, social capital and the legislation of trust,” Master’s thesis, University of Ghent, Belgium, 2007.

- [122] J. Zhang and R. Cohen, "Design of a mechanism for promoting honesty in e-marketplaces," in *22nd Conference on Artificial Intelligence, AI and the Web Track, Vancouver, BC, Canada*, 2007.
- [123] J. Zhang, "Promoting honesty in electronic marketplaces: Combining trust modeling and incentive mechanism design," Ph.D. dissertation, University of Waterloo, Canada, 2009.
- [124] S. Mittal and K. Deb, "Optimal strategies of the iterated prisoner's dilemma problem for multiple conflicting objectives," in *IEEE Symposium on Computational Intelligence and Games, Reno, NV, USA*, 2006, pp. 197–204.
- [125] J. Zhou, J. Wang, R. Liang, and Y. Zhang, "Flexible service analysis based on the prisoner's dilemma of service," in *6th International Conference on Service Systems and Service Management, Xiamen, china*, 2009, pp. 434–437.
- [126] H. Huang, G. Zhu, and S. Jin, "Revisiting trust and reputation in multi-agent systems," in *ISECS International Colloquium on Computing, Communication, Control and Management, Guangzhou, China*, 2008, pp. 424–429.
- [127] L. Mui, "Computational models of trust and reputation: Agents, evolutionary games and social networks," Ph.D. dissertation, Massachusetts Institute of Technology, Boston, MA, USA, 2002.
- [128] M. Momani and S. Challa, "Survey of trust models in different network domains," *International Journal of Ad hoc, Sensor & Ubiquitous Computing*, vol. 1, no. 3, pp. 1–19, 2010.
- [129] T. Y. Chuang, "Trust with social network learning in e-commerce," in *IEEE International Conference on Communications Workshops (ICC), Capetown, South Africa*, 2010, pp. 1–6.
- [130] M. Adamski, A. Arenas, A. Bilas, P. Fragopoulou, V. Georgiev, A. Hevia, G. Jankowski, B. Matthews, N. Meyer, J. Platte, and M. Wilson, "Trust and security in grids: A state of the art," CoreGRID, European Union, Tech. Rep. WHP-0001, 2008.
- [131] A. Gouglidis and I. Mavridis, "A foundation for defining security requirements in grid computing," in *13th Panhellenic Conference on Informatics, Corfu, Greece*, 2009, pp. 180–184.
- [132] L. B. de Oliveira and C. A. Maziero, "A trust model for a group of e-mail servers," *CLEI Electronic Journal*, vol. 11, no. 2, pp. 1–11, 2008.
- [133] Q. Zhang, T. Yu, and K. Irwin, "A classification scheme for trust functions in reputation-based trust management," in *International Workshop on Trust, Security, and Reputation on the Semantic Web, Hiroshima, Japan*, 2004.
- [134] Z. Shen, L. Li, F. Yan, and X. Wu, "Cloud computing system based on trusted computing platform," in *International Conference on Intelligent Computation Technology and Automation (ICICTA), Changsha, China*, vol. 1, 2010, pp. 942–945.

- [135] Z. Shen and Q. Tong, "The security of cloud computing system enabled by trusted computing technology," in *2nd International Conference on Signal Processing Systems (ICSPS), Dalian, China*, vol. 2, 2010, pp. 11–15.
- [136] K. M. Khan and Q. Malluhi, "Establishing trust in cloud computing," *IEEE IT Professional*, vol. 12, no. 5, pp. 20–27, 2010.
- [137] H. Takabi, J. B. D. Joshi, and G. J. Ahn, "Securecloud: Towards a comprehensive security framework for cloud computing environments," in *34th Annual IEEE Computer Software and Applications Conference Workshops, Seoul, South Korea*, 2010, pp. 393–398.
- [138] Z. Song, J. Molina, and C. Strong, "Trusted anonymous execution: A model to raise trust in cloud," in *9th International Conference on Grid and Cooperative Computing (GCC), Nanjing, China*, 2010, pp. 133–138.
- [139] H. Sato, A. Kanai, and S. Tanimoto, "A cloud trust model in a security aware cloud," in *10th IEEE/IPSJ International Symposium on Applications and the Internet, Seoul, South Korea*, 2010, pp. 121–124.
- [140] W. Li, L. Ping, and X. Pan, "Use trust management module to achieve effective security mechanisms in cloud environment," in *International Conference on Electronics and Information Engineering, Kyoto, Japan*, 2010, pp. 14–19.
- [141] T. F. Wang, B. S. Ye, Y. W. Li, and Y. Yang, "Family gene based cloud trust model," in *International Conference on Educational and Network Technology, Qinhuaungdao, China*, 2010, pp. 540–544.
- [142] T. F. Wang, B. S. Ye, Y. W. Li, and L. S. Zhu, "Study on enhancing performance of cloud trust model with family gene technology," in *3rd IEEE International Conference on Computer Science and Information Technology, Chengdu, China*, 2010, pp. 122–126.
- [143] T. S. Somasundaram, B. R. Amarnath, R. Kumar, P. Balakrishnan, K. Rajendar, R. Rajiv, K. Govindarajan, G. R. Britto, E. Mahendran, and B. Madusudhanan, "CARE resource broker: A framework for scheduling and supporting virtual resource management," *Future Generation Computer Systems*, vol. 26, no. 3, pp. 337–347, 2010.
- [144] P. D. Manuel, T. Selve, and M. I. Abd-EI Barr, "Trust management system for grid and cloud resources," in *First International Conference on Advanced Computing (ICAC 2009), Chennai, India*, 2009, pp. 176–181.
- [145] M. Alhamad, T. Dillon, and E. Chang, "SLA-based trust model for cloud computing," in *13th International Conference on Network-Based Information Systems, Takayama, Japan*, 2010, pp. 321–324.
- [146] X. Y. Li, L. T. Zhou, Y. Shi, and Y. Guo, "A trusted computing environment model in cloud architecture," in *Ninth International Conference on Machine Learning and Cybernetics, Qingdao, China*, 2010, pp. 2843–2848.

- [147] Z. Yang, L. Qiao, C. Liu, C. Yang, and G. Wan, "A collaborative trust model of firewall-through based on cloud computing," in *14th International Conference on Computer Supported Cooperative Work in Design, Shanghai, China*, 2010, pp. 329–334.
- [148] J. Fu, C. Wang, Z. Yu, J. Wang, and J. G. Sun, "A watermark-aware trusted running environment for software clouds," in *Fifth Annual ChinaGrid Conference, Guangzhou, China*, 2010, pp. 144–151.
- [149] R. Ranchal, B. Bhargava, L. B. Othmane, L. Lilien, A. Kim, M. Kang, and M. Linderman, "Protection of identity information in cloud computing without trusted third party," in *29th IEEE International Symposium on Reliable Distributed Systems, New Delhi, India*, 2010, pp. 1060–9857.
- [150] L. T. M. Blessing and A. Chakrabarti, *DRM, A Design Research Methodology*. London, UK: Springer, 2009.
- [151] H. Birkhofer, "Are we aware of what we are doing in design research?" in *3rd International Conference on Design Engineering and Science, Pilsen, Czech Republic*, 2014, pp. 1–10.
- [152] A. Wikstrom, "A design process based on visualization," Licentiate Thesis, Malardalen University, Sweden, 2010.
- [153] M. Firdhous, O. Ghazali, and S. Hassan, "Trust management in cloud computing - A critical review," *International Journal on Advances in ICT for Emerging Regions (ICTer)*, vol. 4, no. 2, pp. 24–36, 2011.
- [154] —, "Trust and trust management in cloud computing - a survey," InterNetworks Research Lab, School of Computing, Universiti Utara Malaysia, Sintok, Kedah, Malaysia, Tech. Rep. UUM/CAS/InterNetWorks/TR2011-01, 2011.
- [155] M. Firdhous, S. Hassan, and O. Ghazali, "A comprehensive survey on quality of service implementations in cloud computing," *International Journal of Scientific & Engineering Research*, vol. 4, no. 5, pp. 118–123, 2013.
- [156] M. Firdhous, S. Hassan, O. Ghazali, and M. Mahmuddin, *Cloud Systems in Supply Chains*. Hampshire, UK: Palgrave Macmillan, 2015, ch. Evaluating Cloud System Providers: Models, Methods and Applications, pp. 121–149.
- [157] N. A. Sultan, "Reaching for the cloud: How SMEs can manage," *International Journal of Information Management*, vol. 31, no. 3, pp. 272–278, 2011.
- [158] G. Feng, "Applications of Matlab in mathematical analysis," *Journal of Software*, vol. 6, no. 7, pp. 1225–1229, 2011.
- [159] R. Farrugia, "Modular programming - some lessons learned and benefits gained," in *Computational Science Symposium, Brighton, England*, 2011, pp. 1–6.
- [160] W. J. Palm, *Introduction to Matlab 7 for Engineers*. New York: McGraw Hill, 2004.

- [161] O. Balci, "Verification, validation and accreditation of simulation models," in *29th conference on Winter simulation, Atlanta, GA*, 1997, pp. 135–141.
- [162] T. L. Paez, "Introduction to model validation," in *Conference & Exposition on Structural Dynamics - Model Verification & Validation, Orlando, FL*, 2009, pp. 1–11.
- [163] K. Iwanicki, P. Horban, P. Glazar, and K. Strzelecki, "Bringing modern unit testing techniques to sensornets," *ACM Transactions on Sensor Networks*, vol. 11, no. 2, pp. 1–37, 2015.
- [164] L. Mariani, M. Pezze, and D. Willmor, *Applying Formal Methods: Testing, Performance, and M/E-Commerce*. Berlin Heidelberg: Springer, 2004, ch. Generation of Integration Tests for Self-Testing Components, pp. 337–350.
- [165] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. de Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.
- [166] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: challenges and opportunities," in *7th High Performance Computing and Simulation Conference, Leipzig, Germany*, 2009, pp. 1–11.
- [167] R. Malhotra and P. Jain, "Study and comparison of various cloud simulators available in the cloud computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 9, pp. 347–350, 2013.
- [168] W. Zhao, Y. Peng, F. Xie, and Z. Dai, "Modeling and simulation of cloud computing: A review," in *IEEE Asia Pacific Cloud Computing Congress, Shenzhen, China*, 2012, pp. 20–24.
- [169] F. Fittkau, S. Frey, and W. Hasselbring, "CDOSim: Simulating cloud deployment options for software migration support," in *IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems, Trento, Italy*, 2012, pp. 37–46.
- [170] Y. Jararweh, Z. Alshara, M. Jarrah, M. Kharbutli, and M. N. Alsaleh, "Teachcloud: A cloud computing educational toolkit," *International Journal of Cloud Computing*, vol. 2, no. 2/3, pp. 1–16, 2013.
- [171] G. G. Castane, A. Nunez, and J. Carretero, "iCanCloud: A brief architecture overview," in *10th IEEE International Symposium on Parallel and Distributed Processing with Applications, Madrid, Spain*, 2012, pp. 853–854.
- [172] I. Sriram, "SPECI, a simulation tool exploring cloud-scale data centres," in *First International Conference on Cloud Computing, Beijing, China*, 2009, pp. 381–392.

- [173] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer, "GroudSim: An event-based simulation framework for computational grids and clouds," in *Euro-Par 2010 Parallel Processing Workshops, Ischia, Italy*, 2010, pp. 305–313.
- [174] G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer, "DCSim: A data centre simulation tool," in *IFIP/IEEE International Symposium on Integrated Network Management, Ghent, Belgium*, 2013, pp. 1090–1091.
- [175] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *24th International Conference on Advanced Information Networking and Applications, Perth, Australia*, 2010, pp. 446–452.
- [176] S. K. Garg and R. Buyya, "Networkcloudsim: Modelling parallel applications in cloud simulations," in *4th IEEE/ACM International Conference on Utility and Cloud Computing, Melbourne, Australia*, 2011, pp. 105–113.
- [177] R. Malhotra and P. Jain, "Study and comparison of CloudSim simulators in the cloud computing," *SIJ Transactions on Computer Science Engineering & its Applications*, vol. 1, no. 4, pp. 111–115, 2013.
- [178] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice & Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.
- [179] C. J. Roy, "Review of code and solution verification procedures for computational simulation," *Journal of Computational Physics*, vol. 205, no. 1, pp. 131–156, 2005.
- [180] J. Papcun, "Integrating static code analysis and defect tracking," Master's thesis, Faculty of Informatics, Masaryk University, Brno, Czech Republic, 2014.
- [181] T. Arnold, D. Hopton, A. Leonard, and M. Frost, *Professional Software Testing with Visual Studio 2005 Team System: Tools for Software Developers and Test Engineers*. Hoboken, NJ: Wiley, 2007.
- [182] J. M. Kelif, "Analytical performance model for Poisson wireless networks with pathloss and shadowing propagation," in *Globecom Workshops, Austin, TX*, 2014, pp. 1528–1532.
- [183] I. A. Lawal, A. M. Said, and A. A. Muazu, "Simulation model to improve QoS performance over fixed WiMAX using OPNET," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 6, no. 21, pp. 3933–3945, 2013.
- [184] M. Ivanovich, P. Fitzpatrick, J. Li, M. Beresford, and A. Saliba, "Measuring quality of service in an experimental wireless data network," *Australian Telecommunications, Networks and Applications Conference, Perth, WA, Australia*, 2003.

- [185] M. S. Obaidat and N. A. Boudriga, *Fundamentals of Performance Evaluation of Computer and Telecommunications Systems*. Hoboken, NJ: Wiley, 2010.
- [186] K. Velten, *Mathematical Modeling and Simulation: Introduction for Scientists and Engineers*. Hoboken, NJ: Wiley, 2008.
- [187] A. Doosti and A. M. Ashtiani, "Mathematical modeling: a new approach for mathematics teaching in different levels," in *2nd Meeting of the network of teachers, researchers and undergraduate students of physics and mathematics, Sao Carlos, Brazil*, 2010, pp. 1–9.
- [188] K. Soetaert, T. Petzoldt, and R. W. Setzer, "Solving differential equations in R," *The R Journal*, vol. 2, no. 2, pp. 5–15, 2010.
- [189] J. A. Cabrera and J. P. T. Higgins, "Graphical displays for meta-analysis: An overview with suggestions for practice," *Research Synthesis Methods*, vol. 1, pp. 66–80, 2010.
- [190] R. K. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1991.
- [191] R. Sherwood, G. Gibb, K. K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the testbed?" in *9th USENIX conference on Operating systems design and implementation, Vancouver, BC, Canada*, 2010, pp. 1–14.
- [192] S. B. Lee, E. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, H. Mark, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. Mccanne, and H. Yu, "Improving simulation for network research," Department of Computer Science, University of Southern California, Los Angeles, CA, Technical Report 99-702, 1999.
- [193] E. Weingartner, H. vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *IEEE International Conference on Communications, Dresden, Germany*, 2009, pp. 1–5.
- [194] M. N. Ismail and A. M. Zin, "Comparing the accuracy of network utilization performance between real network and simulation model for Local Area Network (LAN)," *I.J. Communications, Network and System Sciences*, vol. 4, pp. 339–349, 2008.
- [195] R. Shaikh and M. Sasikumar, "Cloud simulation tools: A comparative analysis," *International Journal of Computer Applications*, pp. 11–14, 2013.
- [196] T. Goyal, A. Singh, and A. Agrawal, "Cloudsim: Simulator for cloud computing infrastructure and modeling," *Procedia Engineering*, vol. 38, pp. 3566–3572, 2008.
- [197] R. N. Calheiros, R. Ranjan, C. A. F. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, Technical Report GRIDS-TR-2009-1, 2009.

- [198] A. Nunez, J. L. V. Poletti, A. C. Caminero, G. G. Castane, J. Carretero, and I. M. Llorente, "icancloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, 2012.
- [199] A. Nunez, J. Fernandez, J. D. Garcia, L. Prada, and J. Carretero, "Simcan: A simulator framework for computer architectures and storage networks," in *1st international conference on Simulation Tools and Techniques for Communications, Networks and Systems, Marseille, France*, 2008, pp. 1–8.
- [200] B. Pranggono, D. Alboaneen, and H. Tianfield, *Simulation Technologies in Networking and Communications: Selecting the Best Tool for the Test*. Boca Raton, FL: CRC Press, 2014, ch. Simulation Tools for Cloud Computing, pp. 311–336.
- [201] S. K. Garg, C. S. Yeo, and R. Buyya, "Green cloud framework for improving carbon efficiency of clouds," in *17th International European Conference on Parallel and Distributed Computing, Bordeaux, France*, 2011, pp. 491–502.
- [202] R. Kaur and N. S. Ghumman, "A survey and comparison of various cloud simulators available for cloud environment," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 5, pp. 605–608, 2015.
- [203] U. Sinha and M. Shekhar, "Comparison of various cloud simulation tools available in cloud computing," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 3, pp. 171–176, 2015.
- [204] C. G. Garino, C. Mateos, and E. Pacini, *Cloud Computing and Big Data*. Amsterdam: IOS Press, 2013, ch. ACO-based Dynamic Job Scheduling of Parametric Computational Mechanics Studies on Cloud Computing Infrastructures, pp. 103–122.
- [205] I. Lazar and J. Husar, "Validation of the serviceability of the manufacturing system using simulation," *Journal on Efficiency and Responsibility in Education and Science*, vol. 5, no. 4, pp. 252–261, 2012.
- [206] P. K. Suri and P. Ranjan, "Comparative analysis of software effort estimation techniques," *International Journal of Computer Applications*, vol. 48, no. 21, pp. 12–19, 2012.
- [207] V. Govindarajalu and V. S. S. Kumar, "Integration of simulation modeling and comparison of scheduling methods to minimize the makespan in a printing industry," *Advanced Materials Research*, vol. 488-489, pp. 1119–1124, 2012.
- [208] O. A. C. Mendoza, "A simulator prototype for an ERP system," MSc Thesis, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark, 2005.
- [209] M. Hassan and R. Jain, *High Performance TCP/IP Networking: Concepts, Issues and Solutions*. Upper Saddle River, NJ: Prentice Hall, 2004.

- [210] J. Singh, "Study study of response time in cloud computing response time in cloud computing," *I.J. Information Engineering and Electronic Business*, vol. 5, pp. 36–43, 2014.
- [211] K. Davies. (2013) Sizes for virtual machines. Microsoft Azure. - Accessed on 05/10/2013. [Online]. Available: <https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-size-specs/>
- [212] S. Frey, C. Luthje, and C. Reich, "Key performance indicators for cloud computing SLAs," in *Fifth International Conference on Emerging Network Intelligence, Porto, Portugal*, 2013, pp. 60–64.
- [213] A. Hanemann, A. Liakopoulos, M. Molina, and D. M. Swamy, "A study on network performance metrics and their composition," *Campus-Wide Information Systems*, vol. 23, no. 4, pp. 268–282, 2006.
- [214] F. Sabahi, "Cloud computing reliability, availability and serviceability (RAS): Issues and challenges," *International Journal on Advances in ICT for Emerging Regions*, vol. 4, no. 2, pp. 12–23, 2011.
- [215] J. B. Villegas-Puyod, "Cost effective cloud computing for real-time applications," in *Tenth International Conference on ICT and Knowledge Engineering, Bangkok, Thailand*, 2012, pp. 171–174.
- [216] A. O. Afolabi and E. R. Adagunodo, "Analysis of robustness of a developed e-learning system," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 8, pp. 23–31, 2013.
- [217] H. N. Erb, "A statistical approach for calculating the minimum number of animals needed in research," *Institute for Laboratory Animal Research Journal*, vol. 32, no. 1, pp. 11–16, 1990.
- [218] J. S. Kim and R. J. Dailey, *Biostatistics for Oral Healthcare*. Oxford, UK: Wiley-Blackwell, 2008, ch. Confidence Intervals and Sample Size, pp. 113–126.
- [219] T. Z. Fahidy, "Some applications of bayes' rule in probability theory to electrocatalytic reaction engineering," *International Journal of Electrochemistry*, vol. 2011, pp. 1–5, 2011.
- [220] P. Damien, P. Dellaportas, N. G. Polson, and D. A. Stephens, Eds., *Bayesian Theory and Applications*. Oxford, UK: Oxford University Press, 2013.
- [221] K. S. Kim and Y. S. Choi, "Bayesian network approach to computerized adaptive testing," *International Journal of Smart Home*, vol. 6, no. 3, pp. 75–82, 2012.
- [222] M. Y. Lin, "Bayesian statistics," School of Public Health, Boston University, Boston, MA, Tech. Rep., 2013.
- [223] K. Huang, I. King, and M. R. Lyu, "Learning maximum likelihood semi-naive bayesian network classifier," in *IEEE International Conference on Systems, Man and Cybernetics, Yasmine Hammamet, Tunisia*, 2002, pp. 1–6.

- [224] A. M. Carvalho, A. L. Oliveira, and M. F. Sagot, "Efficient learning of bayesian network classifiers: An extension to the TAN classifier," in *20th Australian Joint Conference on Artificial Intelligence, Gold Coast, Australia*, 2007, pp. 16–25.
- [225] Y. Jing, V. Pavlovic, and J. M. Rehg, "Boosted bayesian network classifiers," *Machine Learning*, vol. 73, no. 2, pp. 155–184, 2008.
- [226] W. M. Grudzewski, I. K. Hejduk, A. Sankowska, and M. Wantuchowicz, *Trust Management in Virtual Work Environments: A Human Factors Perspective*. Boca Raton, FL: CRC Press, 2008.
- [227] M. Carbone, M. Nielsen, and V. Sassone, "A formal model for trust in dynamic networks," in *1st International Conference on Software Engineering and Formal Methods, Brisbane, Australia*, 2003, pp. 54–61.
- [228] B. S. Vanneste, P. Puranam, and T. Kretschmer, "Trust over time in exchange relationships: Meta-analysis and theory," *Strategic Management Journal*, vol. 35, no. 12, pp. 1891–1902, 2014.
- [229] A. Shawish and M. Salama, *Inter-cooperative Collective Intelligence: Techniques and Applications, Studies in Computational Intelligence*. Berlin Heidelberg: Springer, 2014, ch. Cloud Computing: Paradigms and Technologies, pp. 39–67.
- [230] M. Gabel, A. Schuster, and D. Keren, "Communication-efficient distributed variance monitoring and outlier detection for multivariate time series," in *28th IEEE International Parallel & Distributed Processing Symposium, Phoenix, AZ*, 2014, pp. 37–47.
- [231] L. M. Spizman, "Developing statistical based earnings estimates: Median versus mean earnings," *Journal of Legal Economics*, vol. 19, no. 2, pp. 77–82, 2013.
- [232] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized mlp architectures of neural networks," *International Journal of Artificial Intelligence And Expert Systems*, vol. 1, no. 4, pp. 111–122, 2010.
- [233] A. Tsoularis and J. Wallace, "Analysis of logistic growth models," *Mathematical Biosciences*, vol. 179, no. 1, pp. 21–55, 2002.
- [234] Y. Cai and D. Hames, "Minimum sample size determination for generalized extreme value distribution," *Communications in Statistics - Simulation and Computation*, vol. 40, pp. 99–110, 2011.
- [235] M. Firdhous, O. Ghazali, and S. Hassan, "A trust computing mechanism for cloud computing with multilevel thresholding," in *6th International Conference on Industrial and Information Systems, Kandy, Sri Lanka*, 2011, pp. 457–461.
- [236] F. Ikhouane and J. Rodellar, *Systems with Hysteresis: Analysis, Identification and Control Using the Bouc-Wen Model*. John Wiley & Sons, 2007.

- [237] L. M. L. Ny and B. Tuffin, "Pricing a threshold-queue with hysteresis," in *18th IASTED International Conference on Modelling and Simulation (MS 2007)*, Montreal, Canada, 2007.
- [238] J. Urbano, A. P. Rocha, and E. Oliveira, "Trust evaluation for reliable electronic transactions between business partners," in *Agent-Based Technologies and Applications for Enterprise Interoperability*, ser. Lecture Notes in Business Information Processing, K. Fischer, J. Muller, and R. Levy, Eds. Berlin Heidelberg: Springer, 2012, vol. 98, pp. 219–237.
- [239] M. Zaki, "Handover in a wireless local area network (WLAN)," US Patent 7 164 915, January 16, 2007.
- [240] R. Dhaouadi, F. H. Ghorbel, and P. S. Gandhi, "A new dynamic model of hysteresis in harmonic drives," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 6, pp. 1165–1171, 2003.
- [241] E. Marilly, O. Martinot, S. B. Brezetz, and G. Delegue, "Requirements for service level agreement management," in *IEEE Workshop on IP Operations and Management*, Dallas, TX, 2002, pp. 57–62.
- [242] B. Kahanwal and T. P. Singh, "The distributed computing paradigms: P2p, grid, cluster, cloud, and jungle," *International Journal of Latest Research in Science and Technology*, vol. 1, no. 2, pp. 183–187, 2012.
- [243] P. V. Chauhan, "Cloud computing in distributed system," *International Journal of Engineering Research & Technology*, vol. 1, no. 10, pp. 1–8, 2012.
- [244] I. Verma, "Cloud computing: A study of benefits and challenges," *International Journal of advanced studies in Computer Science and Engineering*, vol. 3, no. 7, pp. 14–17, 2014.
- [245] M. Al-Roomi, S. Al-Ebrahim, S. Buqrais, and I. Ahmad, "Cloud computing pricing models: A survey," *International Journal of Grid and Distributed Computing*, vol. 6, no. 5, pp. 93–106, 2013.
- [246] Z. Yao, "Understanding churn in decentralized peer-to-peer networks," PhD Thesis, Texas A&M University, USA, 2009.
- [247] M. Koivisto and A. Urbaczewski, "The relationship between quality of service perceived and delivered in mobile internet communications," *Information Systems and e-Business Management*, vol. 2, pp. 309–323, 2004.
- [248] M. Kafai, B. Bhanu, and L. An, "Cluster-classification bayesian networks for head pose estimation," in *21st International Conference on Pattern Recognition*, Tsukuba, Japan, 2012, pp. 2869–2872.
- [249] H. Amipara, "A survey on cloudsims toolkit for implementing cloud infrastructure," *International Journal of Science Technology & Engineering*, vol. 1, no. 12, pp. 239–243, 2015.

- [250] OMICS Internetaional. (2015) List of countries by Internet connection speeds. Website. OMICS Internetaional. [Online]. Available: http://research.omicsgroup.org/index.php/List_of_countries_by_Internet_connection_speeds

